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

FILEID**UETUNAS00

UE
VO

```
UU      UU  EEEEEEEEEE  TTTTTTTTTT  UU      UU  NN      NN    AAAAAA      SSSSSSSS    000000      000000
UU      UU  EEEEEEEEEE  TTTTTTTTTT  UU      UU  NN      NN    AAAAAA      SSSSSSSS    000000      000000
UU      UU  EE              TT      UU      UU  NN      NN  AA      AA  SS            00    00  00      00
UU      UU  EE              TT      UU      UU  NN      NN  AA      AA  SS            00    00  00      00
UU      UU  EE              TT      UU      UU  NNNN    NN  AA      AA  SS            00  0000  00    0000
UU      UU  EE              TT      UU      UU  NNNN    NN  AA      AA  SS            00  0000  00    0000
UU      UU  EEEEEEEE        TT      UU      UU  NN  NN  NN  AA      AA    SSSSSS      00  00  00  00  00  00
UU      UU  EEEEEEEE        TT      UU      UU  NN  NN  NN  AA      AA    SSSSSS      00  00  00  00  C0  00
UU      UU  EE              TT      UU      UU  NN    NNNN  AAAAAAAAAA            SS  0000    00  0000    00
UU      UU  EE              TT      UU      UU  NN    NNNN  AAAAAAAAAA            SS  0000    00  0000    00
UU      UU  EE              TT      UU      UU  NN      NN  AA      AA            SS  00      00  00      00      ....
UU      UU  EE              TT      UU      UU  NN      NN  AA      AA            SS  00      00  00      00      ....
UUUUUUUUUU  EEEEEEEEEE      TT      UUUUUUUUUU  NN      NN  AA      AA  SSSSSSSS      000000      000000      ....
UUUUUUUUUU  EEEEEEEEEE      TT      UUUUUUUUUU  NN      NN  AA      AA  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
```

64

41
66

64
3A

```
0000     1         .TITLE UETUNAS00 VAX/VMS UETP DEVICE TEST FOR THE UNA
0000     2         .IDENT  'V04-000'
0000     3         .ENABLE SUPPRESSION
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; FACILITY:
0000    31 ;       This module will be distributed with VAX/VMS under the [SYSTEST]
0000    32 ;       account.
0000    33 ;
0000    34 ; ABSTRACT:
0000    35 ;       This test exercises a UNA device using VMS QIO's. It is run as
0000    36 ;       part of the VAX/VMS UETP.
0000    37 ;
0000    38 ; ENVIRONMENT:
0000    39 ;       This program will run in user access mode, with ASTs enabled except
0000    40 ;       during error processing.  This program requires the following
0000    41 ;       privileges and quotas:
0000    42 ;
0000    43 ;--
0000    44 ;
0000    45 ; AUTHOR: Larry Jones,  CREATION DATE: September 03, 1982
0000    46 ;
0000    47 ; MODIFIED BY:
0000    48 ;
0000    49 ;       V03-010 RNH0010         Richard N. Holstein,    08-Aug-1984
0000    50 ;               Extend V03-009 and prevent potential hang.
0000    51 ;
0000    52 ;       V03-009 RNH0009         Richard N. Holstein,    11-Jul-1984
0000    53 ;               Make the test more robust when performing the remote loopback
0000    54 ;               operations - allow for the Ethernet's uncertainty by
0000    55 ;               retransmitting messages.
0000    56 ;
0000    57 ;       V03-008 RNH0006         Richard N. Holstein,    15-Feb-1984
```

```
0000     58 ;                  Take advantage of new UETP message codes.  Fix SSERROR
0000     59 ;                  interaction with RMS_ERROR.  Change UETUNT$T_FILSPC field
0000     60 ;                  to be ASCID instead of ASCIC for better error messages.
0000     61 ;
0000     62 ;          V03-007 RNH0005         Richard N. Holstein,    03-Jan-1984
0000     63 ;                  Bump MIN_MAXBUF to the new minimum value for the DEUNA.
0000     64 ;
0000     65 ;          V03-006 RNH0004         Richard N. Holstein,    19-Dec-1983
0000     66 ;                  Give correct sentinels to Test Controller.  Use LIB$SIGNAL or
0000     67 ;                  $PUTMSG throughout, instead of LIB$PUT_OUTPUT.
0000     68 ;
0000     69 ;          V03-005 RNH0003         Richard N. Holstein,    06-Dec-1983
0000     70 ;                  Give an error message and abort execution if the SYSGEN MAXBUF
0000     71 ;                  parameter is too small.  Change wording of CASE_FAILED message.
0000     72 ;
0000     73 ;          V03-004 RNH0002         Richard N. Holstein,    21-Nov-1983
0000     74 ;                  Use decimal conversion routine for unit numbers.
0000     75 ;
0000     76 ;          V03-003 RNH0001         Richard N. Holstein,    11-Mar-1983
0000     77 ;                  Don't signal ending message in EXIT_HANDLER.
0000     78 ;
0000     79 ;          V03-002 LDJ0002         Larry D. Jones,         28-Feb-1983
0000     80 ;                  Fixed error message assembly error.
0000     81 ;
0000     82 ;          V03-001 LDJ0001         Larry D. Jones,         10-Feb-1983
0000     83 ;                  Upgraded to conform to new template standards.
0000     84 ;**
```

```
                        .SBTTL  Declarations
                ;
                ; INCLUDE FILES:
                ;
                ;       SYS$LIBRARY:LIB.MLB     for general definitions
                ;       SHRLIB$:UETP.MLB        for UETP definitions
                ;
                ; MACROS:
                ;
                        $CHFDEF                         ; Condition handler frame definitions
                        $DCDEF                          ; Device definitions
                        $DIBDEF                         ; Device Information Block
                        $DVIDEF                         ; $GETDVI ITMLST item codes
                        $NMADEF                         ; Network management definition
                        $SHRDEF                         ; Shared messages
                        $SYIDEF                         ; $GETSYI ITMLST item codes
                        $STSDEF                         ; Status return
                        $UETUNTDEF                      ; UETP unit block offset definitions
                        $UETPDEF                        ; UETP
                        $XMDEF                          ; XMDRIVER symbols

                        .MACRO TBL_ENT ENT,VALUE,STRING
                                .=PC1...
                                .WORD   <ENT@15>!VALUE
                                .ADDRESS PC2...
                                PC1...= .
                                .=PC2...
                                .ASCIC  /STRING/
                                PC2...=.
                        .ENDM TBL_ENT
                ;
                ; EQUATED SYMBOLS:
                ;
                ;   Facility number definitions:
00000001                        RMS$_FACILITY = 1

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

                ;   Internal flag bits...:
00000000                        DUMP_MODEV    = 0               ; Set if running in dump mode
                                ASSUME DUMP_MODEV EQ 0          ; This so we can BLBx
00000001                        TEST_OVERV    = 1               ; Set when test is over
00000002                        SAFE_TO_UPDV  = 2               ; Set when it's safe to update UETINIDEV
00000003                        BEGIN_MSGV    = 3               ; Set if "BEGIN" msg has been printed
00000004                        FLAG_SHUTDNV  = 4               ; Set to indicate device should be
                                                                ; shutdown if errors occur
00000005                        ID_FNDV       = 5               ; Remote UNA found
00000006                        BRDCST_NEDEDV= 6                ; Broadcast for another UNA required
00000007                        LONGORSHRTV   = 7                ; Long or short report (if set then short)

```

```
                     0000   143 ;   ...and corresponding masks:
00000001             0000   144         DUMP_MODEM    = 1@DUMP_MODEV
00000002             0000   145         TEST_OVERM    = 1@TEST_OVERV
00000004             0000   146         SAFE_TO_UPDM  = 1@SAFE_TO_UPDV
00000008             0000   147         BEGIN_MSGM    = 1@BEGIN_MSGV
00000010             0000   148         FLAG_SHUTDNM  = 1@FLAG_SHUTDNV
00000020             0000   149         ID_FNDM       = 1@ID_FNDV
00000040             0000   150         BRDCST_NEDEDM= 1@BRDCST_NEDEDV
00000080             0000   151         LONGORSHRTM   = 1@LONGORSHRTV
                     0000   152
                     0000   153 ;   Miscellany:
00000020             0000   154         LC_BITM           = ^X20          ; Mask to convert lower case to upper
00000028             0000   155         REC_SIZE        = 40              ; UETINIDEV.DAT record size
00000084             0000   156         TEXT_BUFFER     = 132             ; Internal text buffer size
00000005             0000   157         XMIT_EFN          = 5             ; EFN used for transmit QIO's
00000006             0000   158         RECV_EFN          = 6             ; EFN used for receive QIO's
00000007             0000   159         RDID_EFN          = 7             ; EFN used for read request
00000004             0000   160         EFN2              = 4             ; EFN used for three minute timer
00000003             0000   161         SS_SYNCH_EFN    = 3               ; Synch miscellaneous system services
0000000F             0000   162         MAX_PROC_NAME   = 15              ; Longest possible process name
0000000A             0000   163         MAX_DEV_DESIG   = 10              ; Longest possible controller name
00000005             0000   164         MAX_UNIT_DESIG  = 5               ; Longest possible unit number
00000007             0000   165         TBL_SIZE          = 7             ; Number of device counters in table
00000C00             0000   166         LIN               = 0             ; Line counter ID number
00000064             0000   167         PRM               = 100           ; Test AST parameter
00000004             0000   168         RECVPOOL_SIZ    = 4               ; Number of preallocated message block
00000003             0000   169         RW_TIME_ID      = 3               ; Timer ID to prevent hang when Read/write
00000001             0000   170         TIME_ID_1       = 1               ; Timer ID to prevent over all hang
00000002             0000   171         TIME_ID_2       = 2               ; Timer ID to use on REQUEST
00000050             0000   172         DIAG_BUF_SIZE   = 80              ; UNA Diagnostic buffer size
000005DC             0000   173         REQUEST_BUFSIZ  = 1500            ; Read request buffer size
000005DC             0000   174         MAX_UNABUF_SIZE= 1500             ; Maximum UNA transfer size
0000000E             0000   175         FORWARD_MSG_SIZ= 14               ; Size of a forward message header
00000080             0000   176         ERRCNT_LEN      = 128             ; Counters size
00000630             0000   177         MIN_MAXBUF      = 1584            ; Minimum value fo SYSGEN's MAXBUF
00000004             0000   178         RETRY_COUNT = 4                   ; Count of retries for remote loopback
                     0000   179
0000002A             0000   180         HARD_ADR          = ^X2A          ; Diagnostic buffer offset to the
                     0000   181                                           ; NI hardware address
0000001A             0000   182         PCSR1             = ^X1A          ; Diagnostic buffer offset to the
                     0000   183                                           ; PCSR1 register contents
0000000F             0000   184         PCSR1$V_XPWR    = 15              ; NI cable connected indicator
0000000E             0000   185         PCSR1$V_ICAB    = 14              ; Internal cable connected indicator
00000008             0000   186         PCSR1$V_SLFTST  = 8               ; Self test error code indicator
00000006             0000   187         PCSR1$S_SLFTST  = 6
                     0000   188
00000090             0000   189         LOOP_PROTOCOL   = ^X90            ; Digital loop protocol number
                     0000   190
                     0000   191 ;   The following definitions are set depending on the device under test.
                     0000   192 ;   Note that we use the UETUNT$T_FILSPC field differently from most tests.  We
                     0000   193 ;   have an ASCID data type instead of an ASCIC data type.
00000000             0000   194         DEVDEP_SIZE     = 0               ; Size of device dependent part of UETUNT
000005DC             0000   195         WRITE_SIZE      = 1500            ; Size of device write buffer
000005DC             0000   196         READ_SIZE       = 1500            ; Size of device read buffer
                     0000   197
                     0000   198         PAGES = <<UETUNT$C_INDSIZ+-       ; Add together all of the pieces...
                     0000   199                  DEVDEP_SIZE+-            ; ...which make up a UETP unit block...
```

```
                    0000    200             WRITE_SIZE+-            ; ...to give to the $EXPREG service below
                    0000    201             READ_SIZE+-
00000007            0000    202             511>7512>
                    0000    203
```

```
                                0000   205          .SBTTL  Read-Only Data
                            00000000   206          .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                0000   207
                                0000   208 ACNT_NAME:                                ; Process name on exit
53 45 54 53 59 53 00000008'010E0000'   0000   209          .ASCID  /SYSTEST/
                                  54   000E
                                000F   210
                                000F   211 TEST_NAME:                                ; This test name
41 4E 55 54 45 55 00000017'010E0000'   000F   212          .ASCID  /UETUNAS00/
                              30 30 53   001D
                                0020   213
                                0020   214 SUPDEV_GBLSEC:                            ; How we access UETSUPDEV.DAT
50 55 53 54 45 55 00000028'010E0000'   0020   215          .ASCID  /UETSUPDEV/
                              56 45 44   002E
                                0031   216
                                0031   217 CONTROLLER:                              ; Logical name of controller
41 4E 4C 52 54 43 00000039'010E0000'   0031   218          .ASCID  /CTRLNAME/
                                 45 4D   003F
                                0041   219
                                0041   220 MODE:                                    ; Run mode logical name
      45 44 4F 4D 00000049'010E0000'   0041   221          .ASCID  /MODE/
                                004D   222
                                004D   223 LOGNAM:                                  ; Logical name for remote address testing
49 4E 54 53 45 54 00000055'010E0000'   004D   224          .ASCID  /TESTNIADR/
                              52 44 41   005B
                                005E   225
                                005E   226 NO_RMS_AST_TABLE:                        ; List of errors for which...
                          00000000'    005E   227          .LONG   RMS$_BLN        ; ...RMS cannot deliver an AST...
                          00000000'    0062   228          .LONG   RMS$_BUSY       ; ...even if one has an ERR= arg
                          00000000'    0066   229          .LONG   RMS$_CDA        ; Note that we can search table...
                          00000000'    006A   230          .LONG   RMS$_FAB        ; ...via MATCHC since <31:16>...
                          00000000'    006E   231          .LONG   RMS$_RAB        ; ...pattern can't be in <15:0>
                          00000014     0072   232 NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                0072   233
                                0072   234 SYS$INPUT:                               ; Name of device from which...
4E 49 24 53 59 53 0000007A'010E0000'   0072   235          .ASCID  /SYS$INPUT/     ; ...the test can be aborted
                              54 55 50   0080
                                0083   236
                                0083   237 INPUT_ITMLST:                           ; $GETDVI arg list for SYS$INPUT
                        0020 0040      0083   238          .WORD   64,DVI$_DEVNAM  ; We need the equivalence name
                 00000053'0000005B'    0087   239          .LONG   BUFFER,BUFFER_PTR
                          00000000     008F   240          .LONG   0               ; Terminate the list
                                0093   241
                                0093   242 REPORT:                                  ; Long or short report logical name
54 52 4F 50 45 52 0000009B'010E0000'   0093   243          .ASCID  /REPORT/
                                00A1   244
                                00A1   245 CS1:                                     ; Device class and type control string
21 20 42 58 32 21 000000A9'010E0000'   00A1   246          .ASCID  /!2XB !2XB /
                              20 42 58 32   00AF
                                00B3   247
                                00B3   248 CS3:                                     ; Device class-only control string
2A 20 42 58 32 21 000000BB'010E0000'   00B3   249          .ASCID  /!2XB **/
                                 2A    00C1
                                00C2   250
                                00C2   251 INIDEV_UPDERR:                           ; Error during exit handler
20 72 6F 72 72 45 000000CA'010E0000'   00C2   252          .ASCID  /Error updating UETINIDEV.DAT./
54 45 55 20 67 6E 69 74 61 64 70 75    00D0
```

```
                                00DC
                                00E7    253
                                00E7    254 DUMP:                                       ; Equivalence string for MODE logical...
      50 4D 55 44 000000EF'010E0000' 00E7    255             .ASCID  /DUMP/              ; ...name to specify dump mode actions
                                00F3    256
                                00F3    257 NIADRWRONG:
64 64 61 20 49 4E 000000FB'010E0000' 00F3    258             .ASCID  /NI address specified by logical name TESTNIADR is illegal length./
69 66 69 63 65 70 73 20 73 73 65 72  0101
61 63 69 67 6F 6C 20 79 62 20 64 65  010D
4E 54 53 45 54 20 65 6D 61 6E 20 6C  0119
65 6C 6C 69 20 73 69 20 52 44 41 49  0125
   2E 68 74 67 6E 65 6C 20 6C 61 67  0131
                                013C    259
                                013C    260 NIADRTESTING:
65 74 6F 6D 65 52 00000144'010E0000' 013C    261             .ASCID  /Remote UNA address testing is !2XB-!2XB-!2XB-!2XB-!2XB-!2XB./
73 73 65 72 64 64 61 20 41 4E 55 20  014A
20 73 69 20 67 6E 69 74 73 65 74 20  0156
32 21 2D 42 58 32 21 2D 42 58 32 21  0162
42 58 32 21 2D 42 58 32 21 2D 42 58  016E
                  2E 42 58 32 21 2D  017A
                                0180    262
                                0180    263 ID_CTRSTR:
61 68 20 41 4E 55 00000188'010E0000' 0180    264             .ASCID  /UNA hardware address is !2XB-!2XB-!2XB-!2XB-!2XB-!2XB./
65 72 64 64 61 20 65 72 61 77 64 72  018E
21 2D 42 58 32 21 20 73 69 20 73 73  019A
58 32 21 2D 42 58 32 21 2D 42 58 32  01A6
2E 42 58 32 21 2D 42 58 32 21 2D 42  01B2
                                01BE    265
                                01BE    266 REMOTE_ID_CTR:
65 74 6F 6D 65 52 000001C6'010E0000' 01BE    267             .ASCID  /Remote UNA address found of !2XB-!2XB-!2XB-!2XB-!2XB-!2XB./
73 73 65 72 64 64 61 20 41 4E 55 20  01CC
32 21 20 66 6F 20 64 6E 75 6F 66 20  01D8
42 58 32 21 2D 42 58 32 21 2D 42 58  01E4
21 2D 42 58 32 21 2D 42 58 32 21 2D  01F0
                  2E 42 58 32        01FC
                                0200    268
                                0200    269 REMOTE_NOTFND:
65 74 6F 6D 65 52 00000208'010E0000' 0200    270             .ASCID  /Remote UNA not detected on the NI./
74 65 64 20 74 6F 6E 20 41 4E 55 20  020E
65 68 74 20 6E 6F 20 64 65 74 63 65  021A
                  2E 49 4E 20        0226
                                022A    271
                                022A    272 BADLEN:                                     ; Message received was wrong length
67 61 73 73 65 4D 00000232'010E0000' 022A    273             .ASCID  /Message #!UL received with wrong length, got !UW, expected !UW./
69 65 63 65 72 20 4C 55 21 23 20 65  0238
6F 72 77 20 68 74 69 77 20 64 65 76  0244
67 20 2C 68 74 67 6E 65 6C 20 67 6E  0250
65 70 78 65 20 2C 57 55 21 20 74 6F  025C
         2E 57 55 21 20 64 65 74 63  0268
                                0271    274
                                0271    275 COUNTER_MSG:
65 7A 2D 6E 6F 4E 00000279'010E0000' 0271    276             .ASCID  /Non-zero UNA counter "!AC" = !UL./
74 6E 75 6F 63 20 41 4E 55 20 6F 72  027F
21 20 3D 20 22 43 41 21 22 20 72 65  028B
                        2E 4C 55     0297
                                029A    277
                                029A    278 CASE_FAILED:
```

```
6F 63 65 72 6E 55 000002A2'010E0000'  029A   279          .ASCID  /Unrecognized counter in NICE message./
74 6E 75 6F 63 20 64 65 7A 69 5E 67   02A8
6D 20 45 43 49 4E 20 6E 69 20 72 65   02B4
                  2E 65 67 61 73 73 65  02C0
                                        02C7   280
                                        02C7   281 TTNAME_ROPTR:                          ; Descriptor for recursive...
                              0000 003F 02C7   282          .WORD   63,0                  ; ...translation of TTNAME
                               0000000A' 02CB   283          .ADDRESS TTNAME
                                        02CF   284
                                        02CF   285 CNTRLCMSG:
65 74 72 6F 62 41 000002D7'010E0000'  02CF   286          .ASCID  \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64   02DD
                  43 2F 4C 52 54 43 20  02E9
                                        02F0   287
                                        02F0   288 NO_CTRLNAME:
6E 6F 63 20 6F 4E 000002F8'010E0000'  02F0   289          .ASCID  /No controller specified./
63 65 70 73 20 72 65 6C 6C 6F 72 74   02FE
                     2E 64 65 69 66 69  030A
                                        0310   290
                                        0310   291 DEAD_CTRLNAME:
20 74 27 6E 61 43 00000318'010E0000'  0310   292          .ASCID  /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
6C 6F 72 74 6E 6F 63 20 74 73 65 74   031E
72 61 6D 20 2C 53 41 21 20 72 65 6C   032A
61 73 75 6E 75 20 73 61 20 64 65 6B   0336
4E 49 54 45 55 20 6E 69 20 65 6C 62   0342
            2E 54 41 44 2E 56 45 44 49  034E
                                        0357   293
                                        0357   294 NOUNIT_SELECTED:
69 6E 75 20 6F 4E 0000035F'010E0000'  0357   295          .ASCID  /No units selected for testing./
20 64 65 74 63 65 6C 65 73 20 73 74   0365
2E 67 6E 69 74 73 65 74 20 72 6F 66   0371
                                        037D   296
                                        037D   297 CAB1_UNPLUGED:
4E 49 4E 52 41 57 00000385'010E0000'  037D   298          .ASCID  /WARNING the NI cable is disconnected./
62 61 63 20 49 4E 20 65 68 74 20 47   038B
6E 6F 63 73 69 64 20 73 69 20 65 6C   0397
                  2E 64 65 74 63 65 6E  03A3
                                        03AA   299
                                        03AA   300 CAB2_UNPLUGED:
4E 55 20 65 68 54 000003B2'010E0000'  03AA   301          .ASCID  /The UNA inter-module cable is unplugged./
75 64 6F 6D 2D 72 65 74 6E 69 20 41   03B8
20 73 69 20 65 6C 62 61 63 20 65 6C   03C4
      2E 64 65 67 67 75 6C 70 6E 75   03D0
                                        03DA   302
                                        03DA   303 ILLEGAL_REC:
61 67 65 6C 6C 49 000003E2'010E0000'  03DA   304          .ASCID  /Illegal record format in file UETINIDEV.DAT!/
72 6F 66 20 64 72 6F 63 65 72 20 6C   03E8
20 65 6C 69 66 20 6E 69 20 74 61 6D   03F4
41 44 2E 56 45 44 49 4E 49 54 45 55   0400
                                 21 54  040C
                                        040E   305
                                        040E   306 TIMEOUT1:
68 73 20 65 6E 4F 00000416'010E0000'  040E   307          .ASCID  /One shot startup timeout./
74 20 70 75 74 72 61 74 73 20 74 6F   041C
                  2E 74 75 6F 65 6D 69  0428
                                        042F   308
                                        042F   309 TIMEOUT2:
```

```
6C 61 6D 72 6F 4E 00000437'010E0000' 042F                .ASCID  /Normal mode startup timeout./
75 74 72 61 74 73 20 65 64 6F 6D 20  043D
         2E 74 75 6F 65 6D 69 74 20 70 0449
                                     0453
                                     0453     TIMEOUT3:
6D 20 6F 68 63 45 0000045B'010E0000' 0453                .ASCID  \Echo mode transmit/receive loop timeout.\
74 69 6D 73 6E 61 72 74 20 65 64 6F  0461
6F 6F 6C 20 65 76 69 65 63 65 72 2F  046D
         2E 74 75 6F 65 6D 69 74 20 70 0479
                                     0483
                                     0483     TIMEOUT4:
6D 20 6F 68 63 45 0000048B'010E0000' 0483                .ASCID  /Echo mode shutdown timeout./
6E 77 6F 64 74 75 68 73 20 65 64 6F  0491
            2E 74 75 6F 65 6D 69 74 20 049D
                                     04A6
                                     04A6     TIMEOUT5:
6D 73 6E 61 72 54 000004AE'010E0000' 04A6                .ASCID  /Transmit to broadcast address startup timeout./
63 64 61 6F 72 62 20 6F 74 20 74 69  04B4
20 73 73 65 72 64 64 61 20 74 73 61  04C0
65 6D 69 74 20 70 75 74 72 61 74 73  04CC
                        2E 74 75 6F  04D8
                                     04DC
                                     04DC     TIMEOUT6:
6D 73 6E 61 72 54 000004E4'010E0000' 04DC                .ASCID  /Transmit to broadcast address timeout./
63 64 61 6F 72 62 20 6F 74 20 74 69  04EA
20 73 73 65 72 64 64 61 20 74 73 61  04F6
            2E 74 75 6F 65 6D 69 74  0502
                                     050A
                                     050A     TIMEOUT7:
6D 73 6E 61 72 54 00000512'010E0000' 050A                .ASCID  /Transmit to broadcast address shutdown timeout./
63 64 61 6F 72 62 20 6F 74 20 74 69  0518
20 73 73 65 72 64 64 61 20 74 73 61  0524
6D 69 74 20 6E 77 6F 64 74 75 68 73  0530
                     2E 74 75 6F 65  053C
                                     0541
                                     0541     TIMEOUT8:
6E 72 65 74 78 45 00000549'010E0000' 0541                .ASCID  /External loopback startup timeout./
20 6B 63 61 62 70 6F 6F 6C 20 6C 61  054F
65 6D 69 74 20 70 75 74 72 61 74 73  055B
                        2E 74 75 6F  0567
                                     056B
                                     056B     TIMEOUT9:
77 20 67 6E 75 48 00000573'010E0000' 056B                .ASCID  /Hung while trying to transmit during the remote loopback test./
20 67 6E 69 79 72 74 20 65 6C 69 68  0579
20 74 69 6D 73 6E 61 72 74 20 6F 74  0585
72 20 65 68 74 20 67 6E 69 72 75 64  0591
61 62 70 6F 6F 6C 20 65 74 6F 6D 65  059D
         2E 74 73 65 74 20 6B 63     05A9
                                     05B1
                                     05B1     TIMEOUT10:
6D 65 73 6E 65 53 000005B9'010E0000' 05B1                .ASCID  /Sensemode timeout./
2E 74 75 6F 65 6D 69 74 20 65 64 6F  05BF
                                     05CB
                                     05CB     TIMEOUT11:
63 20 64 61 65 52 000005D3'010E0000' 05CB                .ASCID  /Read counters timeout./
65 6D 69 74 20 73 72 65 74 6E 75 6F  05D9
                        2E 74 75 6F  05E5
```

```
                                                05E9   338
                                                05E9   339 RETRY_MSG:                              ; Indicate retransmission in rmt loop
55 21 20 6C 6C 41 000005F1'010E0000'            05E9   340         .ASCID  \All !UL messages in the remote loopback test were transmitted\-
69 20 73 65 67 61 73 73 65 6D 20 4C             05F7
65 74 6F 6D 65 72 20 65 68 74 20 6E             0603
65 74 20 6B 63 61 62 70 6F 6F 6C 20             060F
6E 61 72 74 20 65 72 65 77 20 74 73             061B
                  64 65 74 74 69 6D 73          0627
66 73 73 65 63 63 75 73 5F 21 2F 21             062E   341                \!/!_successfully, but only after some were retransmitted.\
6E 6F 20 74 75 62 20 2C 79 6C 6C 75             063A
6D 6F 73 20 72 65 74 66 61 20 79 6C             0646
61 72 74 65 72 20 65 72 65 77 20 65             0652
            2E 64 65 74 74 69 6D 73 6E          065E
                                                0667   342
                                                0667   343 LOOPBACK_FAIL_MSG:                      ; Couldn't loop some messages
65 6D 20 4C 55 21 0000066F'010E0000'            0667   344         .ASCID  \!UL message!%S with size!%S of!#(5UL) were not received\-
74 69 77 20 53 25 21 65 67 61 73 73             0675
66 6F 20 53 25 21 65 7A 69 73 20 68             0681
65 72 65 77 20 29 4C 55 35 28 23 21             068D
65 76 69 65 63 65 72 20 74 6F 6E 20             0699
                                  64            06A5
72 20 65 68 74 20 79 62 5F 21 2F 21             06A6   345                \!/!_by the remote loopback test, even after retransmission.\
61 62 70 6F 6F 6C 20 65 74 6F 6D 65             06B2
65 76 65 20 2C 74 73 65 74 20 6B 63             06BE
72 74 65 72 20 72 65 74 66 61 20 6E             06CA
   2E 6E 6F 69 73 73 69 6D 73 6E 61             06D6
                                                06E1   346
                                                06E1   347 RETRANS_MSG:                            ; Retransmit for remote loopback
74 61 20 4C 55 21 000006E9'010E0000'            06E1   348         .ASCID  /!UL attempt!%S to send message #!UL failed, size !UW, !%T./
20 6F 74 20 53 25 21 74 70 6D 65 74             06EF
65 67 61 73 73 65 6D 20 64 6E 65 73             06FB
64 65 6C 69 61 66 20 4C 55 21 23 20             0707
20 2C 57 55 21 20 65 7A 69 73 20 2C             0713
                  2E 54 25 21                   071F
                                                0723   349
                                                0723   350 LOOPBACK_TMO_MSG:                        ; Timeout during remote loopback
65 74 6F 6D 65 52 0000072B'010E0000'            0723   351         .ASCID  /Remote loopback timeout at !%T for message #!UL./
69 74 20 6B 63 61 62 70 6F 6F 6C 20             0731
54 25 21 20 74 61 20 74 75 6F 65 6D             073D
65 67 61 73 73 65 6D 20 72 6F 66 20             0749
                  2E 4C 55 21 23 20             0755
                                                075B   352
                                                075B   353 OUT_OF_SEQ_MSG:                         ; Wrong or duplicate msg received
74 63 65 70 78 45 00000763'010E0000'            075B   354         .ASCID  /Expected msg #!UL, length !UW, got msg #!UL, length !UW at !%T./
2C 4C 55 21 23 20 67 73 6D 20 64 65             0769
2C 57 55 21 20 68 74 67 6E 65 6C 20             0775
55 21 23 20 67 73 6D 20 74 6F 67 20             0781
55 21 20 68 74 67 6E 65 6C 20 2C 4C             078D
            2E 54 25 21 20 74 61 20 57          0799
                                                07A2   355
                                                07A2   356 ABORT_MSG:                              ; IOSB has SS$_ABORT in remote loopback
                                                07A2   357                                        ; Intentional failure, msg for info only
70 6F 6F 6C 6E 55 000007AA'010E0000'            07A2   358         .ASCID  /Unlooped messages through #!UL have been aborted./
20 73 65 67 61 73 73 65 6D 20 64 65             07B0
4C 55 21 23 20 68 67 75 6F 72 68 74             07BC
61 20 6E 65 65 62 20 65 76 61 68 20             07C8
                  2E 64 65 74 72 6F 62          07D4
```

```
                                                07DB   359
                                                07DB   360 TIME_STAMP_MSG:
30 3D 74 69 6D 58 000007E3'010E0000'            07DB   361         .ASCID  \Xmit=0/recv=1: !UW, length: !4UW, message #!5UL, time: !%T\
57 55 21 20 3A 31 3D 76 63 65 72 2F             07E9
34 21 20 3A 68 74 67 6E 65 6C 20 2C             07F5
20 65 67 61 73 73 65 6D 20 2C 57 55             0801
3A 65 6D 69 74 20 2C 4C 55 3D 21 23             080D
                        54 25 21 20             0819
                                                081D   362
                                                081D   363 TIME_STAMP_INTRO_MSG:
69 20 42 53 4F 49 00000825'010E0000'            081D   364         .ASCID  \IOSB is !XL, !XL; summary of recent messages follows:\
20 3B 4C 58 21 20 2C 4C 58 21 20 73             082B
72 20 66 6F 20 79 72 61 6D 6D 75 73             0837
67 61 73 73 65 6D 20 74 6E 65 63 65             0843
   3A 73 77 6F 6C 6C 6F 66 20 73 65             084F
                                                085A   365
                                                085A   366 PASS_MSG:
66 6F 20 64 6E 45 00000862'010E0000'            085A   367         .ASCID  /End of pass !UL with !UL iterations at !%D./
69 77 20 4C 55 21 20 73 73 61 70 20             0868
61 72 65 74 69 20 4C 55 21 20 68 74             0874
44 25 21 20 74 61 20 73 6E 6F 69 74             0880
                              2E                088C
                                                088D   368
                                                088D   369 ;
                                                088D   370 ;       Table containing the identification for the names of all the
                                                088D   371 ;       UNA device counters.
                                                088D   372 ;
                                                088D   373
                                                088D   374 CNTR_TBL:
                                0000088D        088D   375         PC1... = .
                                000008B7        088D   376         .=.+<TBL_SIZE*6>
                                                08B7   377 TBL_END:
                                                08B7   378
                                                08B7   379 ;
                                                08B7   380 ; *** WARNING *** TBL_SIZE must be modified to reflect the number of entries
                                                08B7   381 ;                  in the following table.
                                                08B7   382 ;
                                                08B7   383
                                000008B7        08B7   384         PC2... = .
                                                08B7   385         .LIST MEB
                                                08B7   386         TBL_ENT LIN,NMA$C_CTLIN_RFL,<packets received in error>
                                0000088D        08B7                       .=PC1...
                                    0426        088D                       .WORD   <LIN@15>!NMA$C_CTLIN_RFL
                                000008B7'       088F                       .ADDRESS PC2...
                                000008B7        0893                       .=PC2...
63 65 72 20 73 74 65 6B 63 61 70 00'            08B7                       .ASCIC  /packets received in error/
72 72 65 20 6E 69 20 64 65 76 69 65             08C3
                                 72 6F          08CF
                                    19          08B7
                                                08D1   387         .NLIST MEB
                                                08D1   388         TBL_ENT LIN,NMA$C_CTLIN_OVR,<receives lost (internal buffer error)>
                                                08F7   389         TBL_ENT LIN,NMA$C_CTLIN_LBE,<receives lost (local buffer error)>
                                                091A   390         TBL_ENT LIN,NMA$C_CTLIN_BSM,<packets transmitted with over 1 collision>
                                                0944   391 ;       TBL_ENT LIN,NMA$C_CTLIN_TFL,<transmit packets aborted>
                                                0944   392         TBL_ENT LIN,NMA$C_CTLIN_CDC,<transmit collision check failures>
                                                0966   393 ;       TBL_ENT LIN,NMA$C_CTLIN_NPT,<no protocol type enabled>
                                                0966   394         TBL_ENT LIN,NMA$C_CTLIN_SBU,<system buffer errors>
```

```
                     097B   395          TBL_ENT LIN,NMA$C_CTLIN_UBU,<user buffer errors>
                     098E   396
                     098E   397 SIZE_TBL:                                ; Table of transfer sizes for echo mode
           00000001  098E   398          .LONG   1                       ; Smallest size transfer
           0000002B  0992   399          .LONG   43                      ; Smallest pad possible
           0000002C  0996   400          .LONG   44                      ; No padding done
           000005DA  099A   401          .LONG   1498                    ; Largest transfer with padding on
           00000004  099E   402          SIZE_TBL_SIZE = <.-SIZE_TBL>/4
                     099E   403
                     099E   404 SIZE_TBL1:                               ; Table of transfer sizes for external
                     099E   405                                          ; loopback mode
           0000002E  099E   406          .LONG   46                      ; Smallest size transfer
           0000007F  09A2   407          .LONG   127                     ; Quarter page-1
           000001FF  09A6   408          .LONG   511                     ; Page-1
           000005DC  09AA   409          .LONG   1500                    ; Largest transfer with padding off
                     09AE   410          ASSUME <<.-SIZE_TBL1>/4> EQ SIZE_TBL_SIZE
                     09AE   411
                     09AE   412 TENSEC:                                  ; 10 seconds delta time
 FFFFFFFF FA0A1F00   09AE   413          .LONG   -10*1000*1000*10,-1
                     09B6   414
                     09B6   415 TWENSEC:                                 ; 20 seconds delta time
 FFFFFFFF F4143E00   09B6   416          .LONG   -10*1000*1000*20,-1
                     09BE   417
                     09BE   418 HALFMIN:                                 ; 30 seconds delta time
 FFFFFFFF EE1E5D00   09BE   419          .LONG   -10*1000*1000*30,-1
                     09C6   420
                     09C6   421 TWOMIN:                                  ; Two minute delta time
 FFFFFFFF B8797400   09C6   422          .LONG   -10*1000*1000*120,-1
                     09CE   423
                     09CE   424 THREEMIN:                                ; 3 minute delta time
 FFFFFFFF 94B62E00   09CE   425          .LONG   -10*1000*1000*180,-1
                     09D6   426
                     09D6   427 UNIT_DESC:                               ; Descriptor used to convert unit #
                            09D6   428          .LONG   5
                            09DA   429          .ADDRESS BUFFER+6
                     09DE   430
                     09DE   431 CONT_DESC:                               ; Descriptor used to convert controller...
                      0028  09DE   432          .WORD   REC_SIZE,0        ; ...from lowercase to uppercase
                            09E2   433          .ADDRESS BUFFER
                     09E6   434
                     09E6   435 FILE:                                    ; Fills in RMS_ERR_STRING
      65 6C 69 66 000009EE'010E000C'  09E6   436          .ASCID  /file/
                     09F2   437
                     09F2   438 RECORD:                                  ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 000009FA'010E0000'  09F2   439          .ASCID  /record/
                     0A00   440
                     0A00   441 RMS_ERR_STRING:                          ; Announces an RMS error
41 21 20 53 4D 52 00000A08'010E0000'  0A00   442          .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53   0A0E
                  44 41 21 20 65 6C 69   0A1A
                     0A21   443
                     0A21   444 PROMPT:
64 20 72 65 6C 6C 6F 72 74 6E 6F 43   0A21   445          .ASCII  /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65   0A2D
                                 20   0A39
                            00000019   0A3A   446          PMTSIZ = .-PROMPT
                     0A3A   447
```

```
                                        0A3A   448 SENSE_ERRMSG:
20 72 6F 72 72 45 00000A42'010E0000'    0A3A   449         .ASCID  /Error in sense mode test /
64 6F 6D 20 65 73 6E 65 73 20 6E 69     0A48
                  20 74 73 65 74 20 65  0A54
                                        0A5B   450
                                        0A5B   451 ERRTEST_MSG:
20 72 6F 72 72 45 00000A63'010E0000'    0A5B   452         .ASCID  /Error in error test /
73 65 74 20 72 6F 72 72 65 20 6E 69     0A69
                              20 74     0A75
                                        0A77   453
                                        0A77   454 ASTPAR_ERRMSG:
20 72 6F 72 72 45 00000A7F'010E0000'    0A77   455         .ASCID  /Error in QIO AST parameter /
70 20 54 53 41 20 4F 49 51 20 6E 69     0A85
         20 72 65 74 65 6D 61 72 61     0A91
                                        0A9A   456
                                        0A9A   457 SLFTST_FAILED:
65 73 20 41 4E 55 00000AA2'010E0000'    0A9A   458         .ASCID  /UNA self test failed with an error code of !XL./
6C 69 61 66 20 74 73 65 74 20 66 6C     0AA8
65 20 6E 61 20 68 74 69 77 20 64 65     0AB4
66 6F 20 65 64 6F 63 20 72 6F 72 72     0AC0
                  2E 4C 58 21 20        0ACC
                                        0AD1   459
                                        0AD1   460 GETSYI_ITMLST:                          ; Check system parameters...
                        104F 0004       0AD1   461         .WORD   4,SYI$_MAXBUF          ; ...to see if we have sufficient MAXBUF
                00000000'00000C50'      0AD5   462         .ADDRESS MAXBUF,0
                        00000000        0ADD   463         .LONG   0                      ; End of ITMLST
                                        0AE1   464
                                        0AE1   465 MAXBUF_MSG:                             ; Tell user to set adequate SYSGEN param
59 53 20 65 68 54 00000AE9'010E0000'    0AE1   466         .ASCID  \The SYSGEN MAXBUF parameter is only !UL.\-
20 46 55 42 58 41 4D 20 4E 45 47 53     0AEF
73 69 20 72 65 74 65 6D 61 72 61 70     0AFB
      2E 4C 55 21 20 79 6C 6E 6F 20     0B07
73 65 74 20 73 69 68 54 5F 21 2F 21     0B11   467                 \!/!_This test requires that the value be at least !UL.\
74 20 73 65 72 69 75 71 65 72 20 74     0B1D
75 6C 61 76 20 65 68 74 20 74 61 68     0B29
73 61 65 6C 20 74 61 20 65 62 20 65     0B35
                  2E 4C 55 21 20 74     0B41
```

```
                            0847   469          .SBTTL  Read/Write Data
                        00000000   470          .PSECT  RWDATA,WRT,NOEXE,PAGE
                            0000   471
                            0000   472 TTCHAN:                                  ; Channel associated with ctrl. term.
                            0000   473          .WORD   0
                            0002   474
                            0002   475 TTNAME_RWPTR:                            ; Descriptor for recursive...
                       0000 0009'  0002   476          .WORD   TTNAME_LEN,0       ; ...translation of TTNAME
                        0000000A'  0006   477          .ADDRESS TTNAME
                            000A   478 TTNAME:
          54 55 50 4E 49 24 53 59 53   000A   479          .ASCII  /SYS$INPUT/
                        00000009   0013   480          TTNAME_LEN=.-TTNAME
                        00000049   0013   481          .BLKB   63-TTNAME_LEN
                            0049   482
                            0049   483 FLAG:                                    ; Miscellaneous flag bits
                            0000   0049   484          .WORD   0                  ; (See Equated Symbols for definitions)
                            004B   485
                            004B   486 FAO_BUF:                                 ; FAO output string descriptor
                       0000 0084   004B   487          .WORD   TEXT_BUFFER,0
                        0000005B'  004F   488          .ADDRESS BUFFER
                            0053   489
                            0053   490 BUFFER_PTR:                             ; Fake .ASCID buffer for misc. strings
                       0000 0084   0053   491          .WORD   TEXT_BUFFER,0      ; A word for length, a word for desc.
                        0000005B'  0057   492          .ADDRESS BUFFER
                            005B   493
                            005B   494 BUFFER:                                  ; FAO output and other misc. buffer
                        000000DF   005B   495          .BLKB   TEXT_BUFFER
                            00DF   496
                            00DF   497 REQUEST_BUF:
                        000006BB   00DF   498          .BLKB   REQUEST_BUFSIZ
                            06BB   499
                            06BB   500 DEVDSC:                                  ; Device name descriptor
                       0000 000A   06BB   501          .WORD   MAX_DEV_DESIG,0
                        000006DA'  06BF   502          .ADDRESS DEV_NAME
                            06C3   503
                            06C3   504
                            06C3   505 PROCESS_NAME:                            ; Process name
53 41 4E 55 000006CB'010E0000'  06C3   506          .ASCID  /UNAS/
                            06CF   507
                        0000000B   06CF   508          PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
                        000006DA   06CF   509          .BLKB   PROCESS_NAME_FREE
                            06DA   510
                            06DA   511 DEV_NAME:                                ; Device name buffer
                        000006E9   06DA   512          .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG
                        0000000F   06E9   513          NAME_LEN = .-DEV_NAME
                            06E9   514
                            06E9   515
                            06E9   516
                            06E9   517 DIB:                                     ; Device Information Block
                       0000 0074   06E9   518          .WORD   DIB$K_LENGTH,0
                        000006F1'  06ED   519          .ADDRESS DIBBUF
                            06F1   520 DIBBUF:
                        00000765   06F1   521          .BLKB   DIB$K_LENGTH
                            0765   522
                            0765   523 ERROR_COUNT:                            ; Cumulative error count at runtime
                        00000000   0765   524          .LONG   0
                            0769   525
```

```
                     0769   526 STATUS:                                 ; Status value on program exit
            00000000 0769   527         .LONG   0
                     076D   528
                     076D   529 QUAD_STATUS:                             ; IO status block for misc sys. svcs.
00000000 00000000    076D   530         .QUAD   0
                     0775   531
                     0775   532 INADDRESS:                               ; $CRMPSC address storage
00000000 00000000    0775   533         .LONG   0,0
                     077D   534 OUTADDRESS:
00000000 00000000    077D   535         .LONG   0,0
                     0785   536
                     0785   537 UNIT_NUMBER:                            ; Current dev unit number for GET_LINE
                0000 0785   538         .WORD   0
                     0787   539
                     0787   540 DEVNAM_LEN:                             ; Current device name length
                0000 0787   541         .WORD   0
                     0789   542
                     0789   543 RANDOM1:                                ; Random word #1
            AAAAAAAA 0789   544         .LONG   ^XAAAAAAAA
                     078D   545
                     078D   546 RANDOM2:                                ; Random word #2
            A72EA72E 078D   547         .LONG   ^XA72EA72E
                     0791   548
                     0791   549 ITERATION:                              ; # of times all tests were executed
            00000000 0791   550         .LONG   0
                     0795   551
                     0795   552 PASS:                                   ; Pass count
            00000000 0795   553         .LONG   0
                     0799   554
                     0799   555 MSG_BLOCK:                              ; Auxiliary $GETMSG info
            0000079D 0799   556         .BLKB   4
                     079D   557
                     079D   558 EXIT_DESC:                              ; Exit handler descriptor
            00000000 079D   559         .LONG   0
            000015FF' 07A1  560         .ADDRESS EXIT_HANDLER
            00000001 07A5   561         .LONG   1
            00000769' 07A9  562         .ADDRESS STATUS
                     07AD   563
                     07AD   564 ARG_COUNT:                              ; Argument counter used by ERROR_EXIT
            00000000 07AD   565         .LONG   0
                     07B1   566
                     07B1   567 ;
                     07B1   568 ; Head of self-relative UETP unit block list.
                     07B1   569 ;
                     07B1   570         .ALIGN QUAD
                     07B8   571
                     07B8   572 UNIT_LIST:                              ; Head of unit block circular list
00000000 00000000    07B8   573         .QUAD   0
                     07C0   574
                     07C0   575 NEW_NODE:                               ; Newly acquired node address
00000000 00000000    07C0   576         .QUAD   0
                     07C8   577
                     07C8   578 RSLLEN:
            00000000 07C8   579         .LONG   0
                     07CC   580
                     07CC   581 RSLBUF:
            00000040 07CC   582         .LONG   64
```

```
                00000704'  07D0    583          .ADDRESS .+4
                00000814   07D4    584          .BLKB   64
                           0814    585
                           0814    586 CVTRSL:
                00000000   0814    587          .LONG   0
                           0818    588
                           0818    589 XE_CHAN:                                ; UNA circuit channel
                    0000   0818    590          .WORD   0
                           081A    591
                           081A    592 XE_CHAN1:                               ; UNA circuit channel for protocol 260
                    0000   081A    593          .WORD   0
                           081C    594
                           081C    595 ARG_LIST:
                00000834   081C    596          .BLKL   6
                           0834    597
                           0834    598 P1BUF:                                  ; P1 Device char buffer
        00000000 00000000  0834    599          .QUAD   0
                           083C    600
                           083C    601 P2_PARAM:
                    0AF1   083C    602          .WORD   NMA$C_PCLI_BUS         ; Buffer size
                000005DC   083E    603          .LONG   MAX_UNABUF_SIZE
                           0842    604
                    0451   0842    605          .WORD   NMA$C_PCLI_BFN         ; Buffer number
                00000005   0844    606          .LONG   5
                           0848    607
                    0B18   0848    608          .WORD   NMA$C_PCLI_PRM         ; Promiscuous mode
                00000001   084A    609          .LONG   NMA$C_STATE_OFF        ; OFF
                           084E    610
                    0B19   084E    611          .WORD   NMA$C_PCLI_MLT         ; Multicast address state
                00000001   0850    612          .LONG   NMA$C_STATE_OFF        ; OFF
                           0854    613
                    0B1B   0854    614          .WORD   NMA$C_PCLI_DCH         ; Data chaining
                00000001   0856    615          .LONG   NMA$C_STATE_OFF        ; OFF
                           085A    616
                    0B1C   085A    617          .WORD   NMA$C_PCLI_CRC         ; CRC
                00000000   085C    618          .LONG   NMA$C_STATE_ON         ; ON
                           0860    619
                    0B1A   0860    620          .WORD   NMA$C_PCLI_PAD         ; Padding
                           0862    621 PAD_SWITCH:
                00000000   0862    622          .LONG   NMA$C_STATE_ON         ; ON
                           0866    623
                    0B0E   0866    624          .WORD   NMA$C_PCLI_PTY         ; Protocol type
                           0868    625 PROTOCOL:
                00000000   0868    626          .LONG   0                      ; 0
                           086C    627
                    0B1F   086C    628          .WORD   NMA$C_PCLI_EKO         ; Echo mode
                           086E    629 ECHO_SWITCH:
                00000000   086E    630          .LONG   NMA$C_STATE_ON         ; ON
                           0872    631
                    0456   0872    632          .WORD   NMA$C_PCLI_CON         ; Controller mode of
                00000000   0874    633          .LONG   NMA$C_LINCN_NOR        ; Normal mode
                           0878    634
                    0B0F   0878    635          .WORD   NMA$C_PCLI_MCA         ; Multicast address
                    0008   087A    636          .WORD   8
                    0001   087C    637          .WORD   NMA$C_LINMC_SET        ; set of
                           087E    638 ADR:
        00 00 02 00 00 AB  087E    639          .BYTE   ^XAB,00,00,02,00,00    ; AB-00-00-02-00-00 which is
```

```
                     0884   640
                     0884   641 P2_DESC:                                    ; Digital remote console address
00000048             0884   642         .LONG   .-P2_PARAM
0000083C'            0888   643         .ADDRESS P2_PARAM
                     088C   644
                     088C   645
                     088C   646 P2_PARAM1:
     0AF1            088C   647         .WORD   NMA$C_PCLI_BUS              ; Buffer size
000005DC             088E   648         .LONG   MAX_UNABUF_SIZE
                     0892   649
     0451            0892   650         .WORD   NMA$C_PCLI_BFN              ; Buffer number
00000005             0894   651         .LONG   5
                     0898   652
     0B18            0898   653         .WORD   NMA$C_PCLI_PRM              ; Promiscuous mode
00000001             089A   654         .LONG   NMA$C_STATE_OFF             ; OFF
                     089E   655
     0B19            089E   656         .WORD   NMA$C_PCLI_MLT              ; Multicast address state
00000001             08A0   657         .LONG   NMA$C_STATE_OFF             ; OFF
                     08A4   658
     0B1B            08A4   659         .WORD   NMA$C_PCLI_DCH              ; Data chaining
00000001             08A6   660         .LONG   NMA$C_STATE_OFF             ; OFF
                     08AA   661
     0B1C            08AA   662         .WORD   NMA$C_PCLI_CRC              ; CRC
00000000             08AC   663         .LONG   NMA$C_STATE_ON              ; ON
                     08B0   664
     0B1A            08B0   665         .WORD   NMA$C_PCLI_PAD              ; Padding
00000000             08B2   666         .LONG   NMA$C_STATE_ON              ; ON
                     08B6   667
     0B0E            08B6   668         .WORD   NMA$C_PCLI_PTY              ; Protocol type
00000260             08B8   669         .LONG   ^X260                       ; 60-02
                     08BC   670
     0B1F            08BC   671         .WORD   NMA$C_PCLI_EKO              ; Echo mode
00000000             08BE   672         .LONG   NMA$C_STATE_ON              ; ON
                     08C2   673
     0456            08C2   674         .WORD   NMA$C_PCLI_CON              ; Controller mode of
00000000             08C4   675         .LONG   NMA$C_LINCN_NOR             ; Loopback mode
                     08C8   676
     0B0F            08C8   677         .WORD   NMA$C_PCLI_MCA              ; Multicast address
     0008            08CA   678         .WORD   8
     0001            08CC   679         .WORD   NMA$C_LINMC_SET             ; set of
00 00 02 00 00 AB    08CE   680         .BYTE   ^XAB,00,00,02,00,00         ; AB-00-00-02-00-00 which is
                     08D4   681                                             ; Digital remote console address
                     08D4   682 P2_DESC1:
00000048             08D4   683         .LONG   .-P2_PARAM1
0000088C'            08D8   684         .ADDRESS P2_PARAM1
                     08DC   685
                     08DC   686 ADR1:                                       ; Read QIO address to fill in
000008EA             08DC   687         .BLKB   14
                     08EA   688
                     08EA   689 ADR2:                                       ; Remote UNA hardware address
000008F8             08EA   690         .BLKB   14
                     08F8   691
                     08F8   692 BRDCST_BUF:
     0000            08F8   693         .WORD   0                           ; Skip
     0002            08FA   694         .WORD   2                           ; Function = Forward
                     08FC   695 SELF_ADR:
00000902             08FC   696         .BLKB   6                           ; Address of self
```

```
                     0001  0902   697          .WORD   1                       ; Function = Reply
                     4A4C  0904   698          .WORD   ^A/LJ/                  ; Receipt number
                 00000926  0906   699          .BLKB   46-<.-BRDCST_BUF>       ; Padding
                 0000002E  0926   700          BRDCST_BUF_LEN = .-BRDCST_BUF
                           0926   701
                           0926   702 SENSE_P1BUF:                            ; P1 buffer for sense mode test
         00000000 00000000 0926   703          .QUAD   0
                           092E   704
                           092E   705 SENSE_P2DESC:                          ; P2 buffer descrip for sense mode test
                 00000090' 092E   706          .LONG   SENSE_P2LEN
                 00000936' 0932   707          .ADDRESS SENSE_P2BUF
                           0936   708
                           0936   709 SENSE_P2BUF:                           ; P2 buffer for sense mode test
                 000009C6  0936   710          .BLKW   <3*24>                  ; 8 quad quad words for dev information
                 00000090  09C6   711 SENSE_P2LEN = .-SENSE_P2BUF             ; P2 buffer length
                           09C6   712
                           09C6   713 ERRTST_P2DESC:                         ; P2 desc for error test
                 00000008' 09C6   714          .LONG   ERRTST_P2LEN
                 000009CE' 09CA   715          .ADDRESS ERRTST_P2BUF
                           09CE   716
                           09CE   717 ERRTST_P2BUF:                          ; P2 buffer for error test
                 000009D6  09CE   718          .BLKQ   1
                 00000008  09D6   719 ERRTST_P2LEN = .-ERRTST_P2BUF
                           09D6   720
                           09D6   721 ERRCOUNT_DESC:                         ; Error counter buffer descrip
                 00000080  09D6   722          .LONG   ERRCNT_LEN
                 000009DE' 09DA   723          .ADDRESS ERRCNT_BUF
                           09DE   724
                           09DE   725 ERRCNT_BUF:                            ; Buffer for error counters
                 00000BDE  09DE   726          .BLKL   ERRCNT_LEN
                           0BDE   727
                           0BDE   728 COUNTER:                               ; Counter value variable
                 00000000  0BDE   729          .LONG   0
                           0BE2   730
                           0BE2   731 END_ADR:                               ; Counter table end address
                 00000000  0BE2   732          .LONG   0
                           0BE6   733
                           0BE6   734 XE_IOSB:                               ; QIO IO status block for transmit
                 00000BEE  0BE6   735          .BLKQ   1
                           0BEE   736
                           0BEE   737 RCV_IOSB:                              ; QIO Io status block for receive
                 00000BF6  0BEE   738          .BLKQ   1
                           0BF6   739
                           0BF6   740 SAVE_STAT:                             ; Set aside status storage
                     0000  0BF6   741          .WORD   0
                           0BF8   742
                           0BF8   743 XMIT_BUF_ADR:                          ; Transmit buffer pointer
                 00000000  0BF8   744          .LONG   0
                           0BFC   745
                           0BFC   746 RECV_BUF_ADR:                          ; Receive buffer pointer
                 00000000  0BFC   747          .LONG   0
                           0C00   748
                           0C00   749 DIAG_BUF:
                 00000C50  0C00   750          .BLKB   DIAG_BUF_SIZE
                           0C50   751
                           0C50   752 MAXBUF:                                ; Value of the SYSGEN MAXBUF param
                 00000C54  0C50   753          .BLKL   1
```

```
                     0C54    754
                     0C54    755 LOOPBACK_ACK:                           ; Flag bytes to signal if some...
00000C58             0C54    756         .BLKB   SIZE_TBL_SIZE           ; ...message was transmitted OK...
                     0C58    757                                         ; ...during the remote loopback test
                     0C58    758
                     0C58    759 SBP:                                    ; Pointer to current slot in...
00000C5C'            0C58    760         .ADDRESS STAMP_BUFFER           ; ...STAMP_BUFFER
                     0C5C    761
                     0C5C    762 STAMP_BUFFER:                           ; Circular list in which to keep time...
                     0C5C    763                                         ; ...stamps for remote loopback msgs
                     0C5C    764         .REPEAT 16                      ; We will have this many "slots", i.e.,
                     0C5C    765                                         ; a history of this many time stamps
                     0C5C    766         ; Note that xxxx_STAMP have dependencies built into them for the
                     0C5C    767         ; contents and length of the slots defined here.
                     0C5C    768         .BLKW                           ; Transm  /receive flag
                     0C5C    769         .BLKW                           ; Message length
                     0C5C    770         .BLKL                           ; Message sequence number (ITERATION)
                     0C5C    771         .BLKQ                           ; Time stamp
                     0C5C    772         .ENDR                           ; End of slot definition
00000100             0D5C    773         TSCLL = .-STAMP_BUFFER          ; Time stamp circular list length
```

```
                     775          .SBTTL  RMS-32 Data Structures
                     776          .ALIGN  LONG
                     777
                     778 SYSIN_FAB:                                  ; Allocate FAB for SYS$INPUT
                     779          $FAB-
                     780          FNM = <SYS$INPUT>
                     781
                     782 SYSIN_RAB:                                  ; Allocate RAB for SYS$INPUT
                     783          $RAB-
                     784          FAB = SYSIN_FAB,-
                     785          ROP = PMT,-
                     786          PBF = PROMPT,-
                     787          PSZ = PMTSIZ,-
                     788          UBF = DEV_NAME,-
                     789          USZ = NAME_LEN
                     790
                     791 INI_FAB:                                  ; Allocate FAB for UETINIDEV
                     792          $FAB-
                     793          FAC = <GET,PUT,UPD>,-
                     794          RAT = CR,-
                     795          SHR = <GET,PUT,UPI>,-
                     796          FNM = <UETINIDEV.DAT>
                     797
                     798 INI_RAB:                                  ; Allocate RAB for UETINIDEV
                     799          $RAB-
                     800          FAB = INI_FAB,-
                     801          RBF = BUFFER,-
                     802          UBF = BUFFER,-
                     803          USZ = REC_SIZE
                     804
                     805 DDB_RFA:                                  ; RFA storage for INI_RAB
00000E8A             806          .BLKB   6
                     807
                     808          .ALIGN  LONG
                     809 SUP_FAB:                                  ; Allocate FAB for UETSUPDEV
                     810          $FAB-
                     811          FAC = GET,-
                     812          SHR = <UPI,GET>,-
                     813          RAT = CR,-
                     814          FOP = UFO,-
                     815          FNM = <UETSUPDEV.DAT>
                     816
                     817 ;
                     818 ; Dummy FAB and RAB to copy to the UETP unit blocks
                     819 ; The following FAB and RAB must be contiguous and in this order!
                     820 ;
                     821
                     822 DUMMY_FAB:
                     823          $FAB
                     824
                     825 DUMMY_RAB:
                     826          $RAB    RSZ = WRITE_SIZE,-
                     827                  USZ = READ_SIZE
```

```
                          0F70   829          .SBTTL  Main Program
                      00000000   830          .PSECT  UNAS,EXE,NOWRT,PAGE
                          0000   831
                          0000   832          .DEFAULT DISPLACEMENT,WORD
                          0000   833
                          0000   834 ;+
                          0000   835 ;       Start up the UNA test.  This entails some overhead necessary to cope
                          0000   836 ;       with both expected and unforseen conditions, figuring out just what
                          0000   837 ;       devices are to be tested, making sure we can test the indicated devices
                          0000   838 ;       and setting up writeable space for each device to be tested.
                          0000   839 ;-
                          0000   840
                      0000  0000   841 .ENTRY UETUNAS00,^M<>                      ; Entry mask
                          0002   842
                6D  13F6'CF   DE  0002   843          MOVAL   SSERROR,(rP)          ; Declare exception handler
                          0007   844          $SETSFM_S ENBFLG = #1                 ; Enable system service failure mode
                          0010   845          $DCLEXH_S DESBLK = EXIT_DESC          ; Declare an exit handler
                          001B   846
                          001B   847          $OPEN   FAB = SYSIN_FAB,-             ; Open SYS$INPUT
                          001B   848                  ERR = RMS_ERROR
                          002A   849          $CONNECT RAB = SYSIN_RAB,-            ; Connect RAB to SYS$INPUT
                          002A   850                   ERR = RMS_ERROR
                    00'   E1  0039   851          BBC     S^#DEV$V_TRM,-        ; BR if SYS$INPUT is NOT a terminal
              22 0D9C'CF      003B   852                  SYSIN_FAB+FAB$L_DEV,10$
                          003F   853          $TRNLOG_S LOGNAM = CONTROLLER,- ; Allow terminal user to specify...
                          003F   854                    RSLLEN = DEVNAM_LEN,- ; ...a logial name...
                          003F   855                    RSLBUF = DEVDSC       ; ...for the controller to test
00000000'8F   50   D1  0058   856          CMPL    R0,#SS$_NORMAL              ; Was a controller specified?
                32   13  005F   857          BEQL    PROC_CONT_NAME              ; BR if it was - go process it
                          0061   858 10$:
                          0061   859          $GET    RAB = SYSIN_RAB,-           ; Read SYS$INPUT...
                          0061   860                  ERR = RMS_ERROR             ; ...for the controller name
            0DCE'CF   B0  0070   861          MOVW    SYSIN_RAB+RAB$W_RSZ,-       ; Save the name length
            0787'CF       0074   862                  DEVNAM_LEN
                  1A  12  0077   863          BNEQ    PROC_CONT_NAME              ; BR if we got something
0769'CF 00000000'8F   D0  0079   864          MOVL    #SS$_BADPARAM,STATUS        ; Save an exit status if not
            02F0'CF   DF  0082   865          PUSHAL  NO_CTRLNAME                 ; Prepare for message...
                  01  DD  0086   866          PUSHL   #1                          ; ...
         00741132 8F   DD  0088   867          PUSHL   #UETP$_TEXT!STS$K_ERROR     ; ...
                  03   DD  008E   868          PUSHL   #3                          ; ...
                14EC   31  0090   869          BRW     ERROR_EXIT                  ; ...to tell of bad setup
                          0093   870
                          0093   871 PROC_CONT_NAME:
    06BB'CF   0787'CF   3C  0093   872          MOVZWL  DEVNAM_LEN,DEVDSC           ; Set the device name length
            06BB'CF   DF  009A   873          PUSHAL  DEVDSC                      ; Make sure...
            06BB'CF   DF  009E   874          PUSHAL  DEVDSC                      ; ...that the specified controller...
 00000000'GF    02   FB  00A2   875          CALLS   #2,G^STR$UPCASE             ; ...is all uppercase for later comaparison
52  06BB'CF    01   C1  00A9   876          ADDL3   #1,DEVDSC,R2                ; Estimate the eventual...
    06C3'CF    52   A0  00AF   877          ADDW2   R2,PROCESS_NAME             ; ...process name length (incl. "_")
               DE   00B4   878          MOVAL   PROCESS_NAME+8-             ; Locate first available byte...
                    00B5   879                  +MAX_PROC_NAME-             ; ...in process name handle...
     50   06CF'CF         00B5   880                  -PROCESS_NAME_FREE,R0       ; ...for device name
            08   C3  00B9   881          SUBL3   #PROCESS_NAME_FREE,-        ; Will the device name fit...
          51   52         00BB   882                  R2,R1                       ; ...in the remaining space?
            08   15  00BD   883          BLEQ    10$                         ; BR if it will
          50   51   C2  00BF   884          SUBL2   R1,R0                           ; Overwrite handle otherwise...
    06C3'CF    0F   B0  00C2   885          MOVW    #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
```

```
                        00C7   886 10$:
            80   5F 8F   90  00C7   887          MOVB    #^A/ /,(R0)+                    ; Separate handle from device name
    60  06DA'CF  06BB'CF   28  00CB   888          MOVC3   DEVDSC,DEV_NAME,(R0)            ; Concatenate handle with device name
                     7E    D4  00D3   889          CLRL    -(SP)                           ; Set the time stamp flag
                000F'CF    DF  00D5   890          PUSHAL  TEST_NAME                       ; Set the test name
                     02    DD  00D9   891          PUSHL   #2                              ; Push the argument count
              00741039 8F   DD  00DB   892          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
          00000000'GF   04   FB  00E1   893          CALLS   #4,G^LIB$SIGNAL                 ; Print the startup message
              0049'CF   08   A8  00E8   894          BISW2   #BEGIN_MSGM,FLAG                ; Set flag so we don't print it again
                             00ED   895          $SETPRN_S PRCNAM = PROCESS_NAME ; Set the process name to UETDMPF00_x
                             00F8   896
                     00'     E1  00F8   897          BBC     S^#DEV$V_TRM,-                  ; BR if SYS$INPUT is NOT a terminal
                66 0D9C'CF         00FA   898                  SYSIN_FAB+FAB$L_DEV,20$
                             00FE   899          $GETDVI_S DEVNAM = SYS$INPUT,-      ; Get the name of...
                             00FE   900                    EFN    = #SS_SYNCH_EFN,- ; ...device which may abort test
                             00FE   901                    ITMLST = INPUT_ITMLST,-
                             00FE   902                    IOSB   = QUAD_STATUS
                45 076D'CF   E9  011A   903          BLBC    QUAD_STATUS,20$                 ; Avoid CTRL/C handler if any error
                             011F   904          $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handler
                             011F   905                    CHAN   = TTCHAN
                             0130   906          $QIOW_S CHAN      = TTCHAN,-        ; Enable CTRL/C AST's...
                             0130   907                  FUNC      = #IO$_SETMODE!IO$M_CTRLCAST,-
                             0130   908                  P1        = CCASTHAND
                06C3'CF      DF  0151   909          PUSHAL  PROCESS_NAME                    ; ...and tell the user...
                     01      DD  0155   910          PUSHL   #1                              ; ...
              0074832B 8F    DD  0157   911          PUSHL   #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
          00000000'GF   03   FB  015D   912          CALLS   #3,G^LIB$SIGNAL                 ; ...
                             0164   913 20$:
```

```
                                0164    915 ;
                                0164    916 ; From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
                                0164    917 ; and unit configuration and lets us know if the setup to run this test was
                                0164    918 ; done correctly.
                                0164    919 ;
                                0164    920         $OPEN   FAB = INI_FAB,-         ; Open file 'UETINIDEV.DAT"
                                0164    921                 ERR = RMS_ERROR
                                0173    922         $CONNECT RAB = INI_RAB,-        ; Connect the RAB and FAB
                                0173    923                  ERR = RMS_ERROR
                                0182    924         $MGBLSC_S   INADR = INADDRESS,- ; Connect to UETSUPDEV global section
                                0182    925                   RETADR = OUTADDRESS,-
                                0182    926                   GSDNAM = SUPDEV_GBLSEC,-
                                0182    927                    FLAGS = #SEC$M_EXPREG
          00000000'8F   50  D1  01A1    928         CMPL    R0,#SS$_NOSUCHSEC       ; Was the section already there?
                        37  12  01A8    929         BNEQ    30$                     ; BR if it was...
                            01AA    930         $OPEN   FAB = SUP_FAB,-         ; ...else open "UETSUPDEV.DAT"
                            01AA    931                 ERR = RMS_ERROR
                            01B9    932         $CRMPSC_S CHAN = SUP_FAB+FAB$L_STV,- ; Create the global section
                            01B9    933                   INADR = INADDRESS,-
                            01B9    934                  RETADR = OUTADDRESS,-
                            01B9    935                  GSDNAM = SUPDEV_GBLSEC,-
                            01B9    936                   FLAGS = #SEC$M_EXPREG!SEC$M_GBL
                            01E1    937 30$:
      56  0781'CF   077D'CF   C3    01E1    938         SUBL3   OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length
                            01E9    939
                            01E9    940 FIND_IT:
                            01E9    941         $GET    RAB = INI_RAB,-         ; Get the first record
                            01E9    942                 ERR = RMS_ERROR
                09DE'CF     DF  01F8    943         PUSHAL  CONT_DESC               ; Make sure...
                09DE'CF     DF  01FC    944         PUSHAL  CONT_DESC               ; ...that the controller name...
          00000000'GF       02  FB  0200    945         CALLS   #2,G^STR$UPCASE         ; ...is all uppercase letters
           005B'CF    44  8F   91  0207    946         CMPB    #^A/D/,BUFFER           ; Is this a DDB?
                        27  13  020D    947         BEQL    10$                     ; Go on if not
           005B'CF    45  8F   91  020F    948         CMPB    #^A/E/,BUFFER           ; Is this the end of the file?
                        D2  12  0215    949         BNEQ    FIND_IT                 ; Continue on if not
                  06BB'CF   DF  0217    950         PUSHAL  DEVDSC                  ; Push device not supported message
                  06C3'CF   DF  021B    951         PUSHAL  PROCESS_NAME            ; Parameters on the stack
                        02  DD  021F    952         PUSHL   #2
                00748333  8F  DD  0221    953         PUSHL   #UETP$_DENOSU
                        02  F0  0227    954         INSV    #STS$K_ERROR,-
                        00      0229    955                 #STS$V_SEVERITY,-
                     6E 03      022A    956                 #STS$S_SEVERITY,(SP)    ; Set the severity code...
                0769'CF 6E  D0  022C    957         MOVL    (SP),STATUS             ; ...and save it as the exit status
                        04  DD  0231    958         PUSHL   #4
                    1349    31  0233    959         BRW     ERROR_EXIT              ; Exit in error
                            0236    960 10$:
  06DA'CF   0061'CF   0787'CF   29  0236    961         CMPC    DEVNAM_LEN,BUFFER+6,DEV_NAME ; Is this the right controller?
                        A7  12  0240    962         BNEQ    FIND_IT                 ; BR if not
       0E84'CF   0E50'CF   06  28  0242    963         MOVC3   #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the Record File Address
            005F'CF    54  8F   91  024A    964         CMPB    #^A/T/,BUFFER+4         ; Can we test this controller?
                        33  13  0250    965         BEQL    FOUND_IT                ; BR if we can...
                            0252    966         $FAO_S  CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if we can't
                            0252    967                 OUTLEN = BUFFER_PTR,-
                            0252    968                 OUTBUF = FAO_BUF,-
                            0252    969                 P1     = #DEVDSC
        0769'CF     00000000'8F DO  026B    970         MOVL    #SS$_BADPARAM,STATUS    ; Set return status
                    0053'CF DF  0274    971         PUSHAL  BUFFER_PTR              ; ...
```

```
                01   DD  0278   972          PUSHL   #1                              ; ...
        00741132 8F   DD  027A   973          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; ...
                03   DD  0280   974          PUSHL   #3                              ; ...
              12FA   31  0282   975          BRW     ERROR_EXIT                      ; We can't test what we can't test
                         0285   976
                         0285   977 FOUND_IT:
                         0285   978          $GET    RAB = INI_RAB,-                 ; Get a record
                         0285   979                  ERR = RMS_ERROR
            09DE'CF   DF  0294   980          PUSHAL  CONT_DESC                       ; Make sure...
            09DE'CF   DF  0298   981          PUSHAL  CONT_DESC                       ; ...that this line...
  00000000'GF   02   FB  029C   982          CALLS   #2,G^STR$UPCASE                 ; ...is all uppercase letters
   005B'CF   55 8F   91  02A3   983          CMPB    #^A/U/,BUFFER                   ; Is this a UCB?
                24   13  02A9   984          BEQL    30$                             ; BR if it is
   005B'CF   44 8F   91  02AB   985          CMPB    #^A/D/,BUFFER                   ; Is this a DDB?
                19   13  02B1   986          BEQL    20$                             ; BR if yes
   005B'CF   45 8F   91  02B3   987          CMPB    #^A/E/,BUFFER                   ; Is this the end?
                11   13  02B9   988          BEQL    20$                             ; BR if yes
                         02BB   989 10$:
            03DA'CF   DF  02BB   990          PUSHAL  ILLEGAL_REC                     ; Then this is an error in the record
                01   DD  02BF   991          PUSHL   #1                              ; Push the error message
        00741132 8F   DD  02C1   992          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; Push the signal name
                03   DD  02C7   993          PUSHL   #3                              ; Push the temp arg count
              12B3   31  02C9   994          BRW     ERROR_EXIT                      ; Finish for good
                         02CC   995 20$:
              0128   31  02CC   996          BRW     ALL_SET                         ; Found DDB or END
                         02CF   997 30$:
   005F'CF   54 8F   91  02CF   998          CMPB    #^A/T/,BUFFER+4                 ; Is the unit testable?
                AE   12  02D5   999          BNEQ    FOUND_IT                        ; BR if not
                01   DD  02D7  1000          PUSHL   #1                              ; Flag to ignore blanks when converting
                02   DD  02D9  1001          PUSHL   #2                              ; Set byte size of results
            0785'CF   DF  02DB  1002          PUSHAL  UNIT_NUMBER                     ; Set address to receive word
            09D6'CF   DF  02DF  1003          PUSHAL  UNIT_DESC                       ; Push string address
     00000000'GF   04   FB  02E3  1004          CALLS   #4,G^OTS$CVT_TI_L               ; Convert ASCII unit # to decimal
                   CE 50   E9  02EA  1005          BLBC    R0,10$                          ; Don't allow bogus unit to pass
                05   20   3B  02ED  1006          SKPC    #^A/ /,#MAX_UNIT_DESIG,-        ; Find out where unit number really is
            0061'CF         02F0  1007                  BUFFER+6
                50   D7  02F3  1008          DECL    R0                              ; Units must all be at least one digit
          61    50   30   3B  02F5  1009          SKPC    #^A/0/,R0,(R1)                  ; Skip leading zeroes on the unit
                50   D6  02F9  1010          INCL    R0                              ; Compensate for DECL above
  06BB'CF   0787'CF   50   A1  02FB  1011          ADDW3   R0,DEVNAM_LEN,DEVDSC            ; Calculate device'unit string length
          52    0787'CF   3C  0303  1012          MOVZWL  DEVNAM_LEN,R2                   ; Offset to unit number in DEVDSC
       06DA'C2  61   50   28  0308  1013          MOVC3   R0,(R1),DEV_NAME(R2)            ; Append unit number to device
                         030E  1014          $GETDEV_S DEVNAM = DEVDSC,-             ; Get the device characteristics
                         030E  1015                    PRIBUF = DIB
          57    06F5'CF   9A  0323  1016          MOVZBL  DIBBUF+DIB$B_DEVCLASS,R7 ; Save the device class
          58    06F6'CF   9A  0328  1017          MOVZBL  DIBBUF+DIB$B_DEVTYPE,R8 ; Save the device type
                         032D  1018          $FAO_S  CTRSTR = CS1,-
                         032D  1019                  OUTBUF = FAO_BUF,-
                         032D  1020                  P1     = R7,-
                         032D  1021                  P2     = R8                     ; Make it into a string
077D'DF   56   005B'CF   06   39  0342  1022          MATCHC  #6,BUFFER,R6,@OUTADDRESS ; Find the device c'ass and type
                1E   13  034B  1023          BEQL    40$                             ; BR if it was found
                         034D  1024          $FAO_S  CTRSTR = CS3,-                  ; Try for full class support
                         034D  1025                  OUTBUF = FAO_BUF,-
                         034D  1026                    P1 = R7
077D'DF   56   005B'CF   06   39  0360  1027          MATCHC  #6,BUFFER,R6,@OUTADDRESS ; Find the device class only
                0D   12  0369  1028          BNEQ    50$                             ; BR if not found
```

```
                     036B  '029 40$:
     55  000F'CF    9A  036B  1030          MOVZBL  TEST_NAME,R5            ; Get the test name length
0017'CF  63    55    29  0370  1031          CMPC3   R5,(R3),TEST_NAME+8     ; Are we the right test?
               1F    13  0376  1032          BEQL    60$                     ; BR if yes
                     0378  1033 50$:
         06BB'CF    DF  0378  1034          PUSHAL  DEVDSC                  ; Push device not supported message
         06C3'CF    DF  037C  1035          PUSHAL  PROCESS_NAME            ; Parameters on the stack
               02   DD  0380  1036          PUSHL   #2                      ; Push the argument count
 00748333      8F   DD  0382  1037          PUSHL   #UETP$_DENOSU
               02   F0  0388  1038          INSV    #STS$K_ERROR,-
               00       038A  1039                  #STS$V_SEVERITY,-
           6E  03       038B  1040                  #STS$S_SEVERITY,(SP)    ; Set the severity code...
    0769'CF    6E  D0  038D  1041          MOVL    (SP),STATUS             ; ...and save it as the exit status
               04  DD  0392  1042          PUSHL   #4                      ; Push the partial arg count...
         11E8  31  0394  1043          BRW     ERROR_EXIT                  ; ...and split this scene
```

```
0397  1045 ;+
0397  1046 ; The following code dynamically allocates enough memory for a unit block,
0397  1047 ; a device dependent parameter area and I/O buffers. The unit block is inserted
0397  1048 ; into the queue header UNIT_LIST.  It then initializes the unit block.
0397  1049 ; A comment indicates where the device dependent parameters should be
0397  1050 ; initialized.  The unit block format is as follows:
0397  1051 ;
0397  1052 ;                                +---------------+ -------
0397  1053 ;       UETUNT$L_FLINK           !               !       ^
0397  1054 ;                                +---------------+       !
0397  1055 ;       UETUNT$L_BLINK           !               !       !
0397  1056 ;                                +-----------+---+       !
0397  1057 ;       UETUNT$B_TYPE                        !   !       !
0397  1058 ;                                    +-------+---+       !
0397  1059 ;       UETUNT$W_SIZE                !   !   !    contains DEVDEP_SIZE + UETUNT$C_INDSIZ
0397  1060 ;                                +---+-------+           !
0397  1061 ;       UETUNT$B_FLAGS           !   !                   !
0397  1062 ;                                +---+   +-------+       !
0397  1063 ;       UETUNT$W_CHAN                !   !   !           !
0397  1064 ;                                +-------+-------+       !
0397  1065 ;       UETUNT$W_FUNC            !   !   !               !
0397  1066 ;                                +-------+-------+       +----- UETUNT$C_SIZE
0397  1067 ;       UETUNT$L_ITER            !               !       !
0397  1068 ;                                +---------------+       !
0397  1069 ;       UETUNT$T_FILSPC          !               !       .
0397  1070 ;                                !/\/\/\/\/\/\/\/!
0397  1071 ;                                NAM$C_MAXRSS bytes      .
0397  1072 ;                                !/\/\7\/\/\/\/\/.
0397  1073 ;                                !               !       .
0397  1074 ;       UETUNT$K_FAB             !---------------!
0397  1075 ;                                !               !
0397  1076 ;                                !/\/\/\/\/\/\/\/.
0397  1077 ;                                 FAB$C_BLN bytes
0397  1078 ;                                !/\/\/\/\/\/\/\/.
0397  1079 ;                                !               !       .
0397  1080 ;       UETUNT$K_RAB             +---------------+       .
0397  1081 ;                                !               !       .
0397  1082 ;                                !\/\/\/\/\/\/\/\!       .
0397  1083 ;                                 RAB$C_BLN bytes        .
0397  1084 ;                                !\/\/\7\/\/\/\/\!       .
0397  1085 ;                                !               !       v
0397  1086 ;       UETUNT$K_DEVDEP          +---------------+ -------
0397  1087 ;                                !               !       ^
0397  1088 ;                                !\/\/\/\/\/\/\/\!       !
0397  1089 ;                                  user defined          +----- DEVDEP_SIZE
0397  1090 ;                                !\/\/\/\/\/\/\/\!       !
0397  1091 ;                                !               !       v
0397  1092 ;                                +---------------+ -------
0397  1093 ;     READ/WRITE buffers         !               !       ^
0397  1094 ;                                !\/\/\/\/\/\/\/\!       !
0397  1095 ;                                  user defined          +----- WRITE_SIZE and READ_SIZE
0397  1096 ;                                !/\/\/\/\/\/\/\!        !
0397  1097 ;                                !               !       v
0397  1098 ;                                +---------------+ -------
0397  1099 ;-
```

```
                                0397  1101 60$:
                                0397  1102          $EXPREG_S PAGCNT = #PAGES,-      ; Get a new node of demand zero memory
                                0397  1103                    RETADR = NEW_NODE
07B8'CF    07C0'DF    5D    03A8  1104          INSQTI  @NEW_NODE,UNIT_LIST      ; Put the new node in the unit list
     56    07C0'CF    D0    03AF  1105          MOVL    NEW_NODE,R6              ; Save a copy of its address
       08 A6    01    90    03B4  1106          MOVB    #1,UETUNT$B_TYPE(R6)     ; Set the structure type
          01A4 8F    B0    03B8  1107          MOVW    #UETUNT$C_INDSIZ+DEVDEP_SIZE,-
            09 A6          03BC  1108                  UETUNT$W_SIZE(R6)        ; Set the structure size
  14 A6    06BB'CF    D0    03BE  1109          MOVL    DEVDSC,UETUNT$T_FILSPC(R6) ; Set the device name size...
            1C A6    DE    03C4  1110          MOVAL   UETUNT$T_FILSPC+8(R6),-   ; ...and pointer to the name
            18 A6          03C7  1111                  UETUNT$T_FILSPC+4(R6)
06BF'DF    06BB'CF    28    03C9  1112          MOVC3   DEVDSC,@DEVDSC+4,-
            1C A6          03D0  1113                  UETUNT$T_FILSPC+8(R6)    ; Save the device name
          0094 8F    28    03D2  1114          MOVC3   #FAB$C_BLN+RAB$C_BLN,-
0110 C6    0EDC'CF          03D6  1115                  DUMMY_FAB,UETUNT$C_FAB(R6) ; Save a FAB and a RAB away
     57    0110 C6    DE    03DC  1116          MOVAL   UETUNT$K_FAB(R6),R7      ; Save the FAB address
     58    0160 C6    DE    03E1  1117          MOVAL   UETUNT$K_RAB(R6),R8      ; Save the RAB address
       3C A8    57    D0    03E6  1118          MOVL    R7,RAB$L_FAB(R8)         ; Set the FAB address in the RAB
            14 A6    90    03EA  1119          MOVB    UETUNT$T_FILSPC(R6),-
            34 A7          03ED  1120                  FAB$B_FNS(R7)            ; Set the FNS field in the FAB
            1C A6    DE    03EF  1121          MOVAL   UETUNT$T_FILSPC+8(R6),-
            2C A7          03F2  1122                  FAB$L_FNA(R7)            ; Set the FNA field in the FAB
                           03F4  1123 ;
                           03F4  1124 ; Set the device dependent parameters in here
                           03F4  1125 ;
              FE8E    31   03F4  1126          BRW     FOUND_IT                 ; Do the next UCB
```

```
                        03F7  1128          .SBTTL  Test the DEUNA
                        03F7  1129
                        03F7  1130 ;
                        03F7  1131 ; Arrive here when we have the device configuration.  In normal or loop forever
                        03F7  1132 ; mode, set a timer far enough in the future such that we can do a reasonable
                        03F7  1133 ; set of tests before the timer expires, but if our device gets hung, the
                        03F7  1134 ; program won't waste too much time before noticing.  Let one-shot mode be a
                        03F7  1135 ; special case.
                        03F7  1136 ;
                        03F7  1137 ALL_SET:
          07B8'CF   D5  03F7  1138          TSTL    UNIT_LIST                       ; Anything to test?
                1A  12  03FB  1139          BNEQ    10$                             ; BR if yes
          0357'CF   DF  03FD  1140          PUSHAL  NOUNIT_SELECTED                 ; Else set up the error message...
                01  DD  0401  1141          PUSHL   #1                              ; ...argument count...
           00741132 8F  DD  0403  1142          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; ...signal name...
                03  DD  0409  1143          PUSHL   #3                              ; ...and parameter count
0769'CF    00000000'8F  D0  040B  1144          MOVL    #SS$_BADPARAM,STATUS            ; Set return status
                1168  31  0414  1145          BRW     ERROR_EXIT                      ; ...and give up, complaining
                        0417  1146 10$:
          0049'CF   04  A8  0417  1147          BISW2   #SAFE_TO_UPDM,FLAG              ; OK safe to update UETINIDEV.DAT now
                        041C  1148
                        041C  1149          $GETSYI_S ITMLST = GETSYI_ITMLST ; Check for an adequate SYSGEN MAXBUF
0C50'CF    00000630 8F  D1  0431  1150          CMPL    #MIN_MAXBUF,MAXBUF              ; Is it .GE. our minimum?
                33  15  043A  1151          BLEQ    12$                             ; BR if it is
                        043C  1152          $FAO_S  CTRSTR = MAXBUF_MSG,-           ; Complain if it isn't...
                        043C  1153                  OUTBUF = FAO_BUF,-
                        043C  1154                  OUTLEN = BUFFER_PTR,-
                        043C  1155                  P1     = MAXBUF,-
                        043C  1156                  P2     = #MIN_MAXBUF
          0769'CF   00' 9A  0459  1157          MOVZBL  S^#SS$_BADPARAM,STATUS
               0053'CF  DF  045E  1158          PUSHAL  BUFFER_PTR
                01  DD  0462  1159          PUSHL   #1
           00741134 8F  DD  0464  1160          PUSHL   #UETP$_TEXT!STS$K_SEVERE
                03  DD  046A  1161          PUSHL   #3
              1110  31  046C  1162          BRW     ERROR_EXIT                      ; ...and give up altogether
                        046F  1163 12$:
                        046F  1164          $ASSIGN_S -                             ; Assign channel to the device
                        046F  1165                  DEVNAM = DEVDSC,-
                        046F  1166                  CHAN = XE_CHAN
                        0480  1167
              14 50     E9  0480  1168          BLBC    R0,15$                          ; BR if no failure
                        0483  1169
                        0483  1170          $ASSIGN_S -                             ; Assign channel to the device
                        0483  1171                  DEVNAM = DEVDSC,-
                        0483  1172                  CHAN = XE_CHAN1
                        0494  1173
              26 50     E8  0494  1174          BLBS    R0,20$                          ; BR if no failure
                        0497  1175 15$:
          0769'CF   50  D0  0497  1176          MOVL    R0,STATUS                       ; Save the failure status
               0765'CF  D6  049C  1177          INCL    ERROR_COUNT                     ; Bump the error count
               0769'CF  DD  04A0  1178          PUSHL   STATUS                          ; Push the error code...
               0769'CF  DD  04A4  1179          PUSHL   STATUS
               06BB'CF  DF  04A8  1180          PUSHAL  DEVDSC                          ; ...and the device designation...
               000F'CF  DF  04AC  1181          PUSHAL  TEST_NAME                       ; ...and the test name...
                03  DD  04B0  1182          PUSHL   #3                              ; ...and the arg count...
           0074819A 8F  DD  04B2  1183          PUSHL   #UETP$_DEUNUS!STS$K_ERROR       ; ...and the signal name...
                06  DD  04B8  1184          PUSHL   #6                              ; ...and the total argument count...
```

```
                10C2   31   04BA  1185          BRW     ERROR_EXIT                ; ...and bail out completely
                            04BD  1186 20$:
                            04BD  1187          $TRNLOG_S LOGNAM = REPORT,-       ; Get the report mode
                            04BD  1188                    RSLLEN = BUFFER_PTR,-
                            04BD  1189                    RSLBUF = FAO_BUF
     005B'CF    20   8A     04D6  1190          BICB2   #LC_BITM,BUFFER           ; Convert to upper case
   005B'CF    53 8F   91     04DB  1191          CMPB    #^A/S/,BUFFER             ; Is this short report
                07   12     04E1  1192          BNEQ    30$                       ; BR if not
 0049'CF    0080 8F   A8     04E3  1193          BISW2   #LONGORSHRTM,FLAG         ; Set short report bit
                            04EA  1194 30$:
                            04EA  1195          $TRNLOG_S LOGNAM = MODE,-         ; Get the run mode
                            04EA  1196                    RSLLEN = BUFFER_PTR,-
                            04EA  1197                    RSLBUF = FAO_BUF
          0053'CF   DF      0503  1198          PUSHAL  BUFFER_PTR
          0053'CF   DF      0507  1199          PUSHAL  BUFFER_PTR
 00000000'GF    02   FB     050B  1200          CALLS   #2,G^STR$UPCASE
00EB'DF   00E7'CF   39      0512  1201          MATCHC  DUMP,@DUMP+4,-            ; Check for running in DUMP mode
005B'CF   0053'CF           0519  1202                  BUFFER_PTR,BUFFER
                05   12     051F  1203          BNEQ    35$                       ; BR if we are not
     0049'CF    01   A8     0521  1204          BISW2   #DUMP_MODEM,FLAG          ; Remember it if we are
                            0526  1205 35$:
   005B'CF    4F 8F   91     0526  1206          CMPB    #^A/O/,BUFFER             ; Is this a one shot?
                03   13     052C  1207          BEQL    40$                       ; BR if not
                0158   31   052E  1208          BRW     TIME_IT
                            0531  1209 40$:     ; Fall into one shot testing
```

```
                     0531  1211 ;
                     0531  1212 ; Note: A startup is all that is needed to ONE SHOT test the UNA because
                     0531  1213 ;       the driver performs an XE$C_DG_STEST diagnostic function on the
                     0531  1214 ;       UNA in a startup sequence and the PCSR1 contents are checked by
                     0531  1215 ;       the device test to make sure that the self test function completed
                     0531  1216 ;       successfully. The XE$C_DG_STEST function performs a read write
                     0531  1217 ;       verification of the UNA.
                     0531  1218 ;
                     0531  1219
                     0531  1220          $SETIMR_S -                         ; Set up ten second timer
                     0531  1221                  DAYTIM = TWENSEC,-          ; to prevent hang
                     0531  1222                  ASTADR = TIME_ERR_OUT,-
                     0531  1223                  REQIDT = #TIMEOUTT
                     0548  1224
                     0548  1225          $SETSFM_S ENBFLG = #0               ; Disable system service failure mode
                     0551  1226
                     0551  1227          $QIOW_S FUNC = #IO$_SETCHAR!IO$M_CTRL!IO$M_STARTUP,-
                     0551  1228                  CHAN = XE_CHAN,-
                     0551  1229                  IOSB = XE_IOSB,-
                     0551  1230                  P2   = #P2_DESC,-
                     0551  1231                  P6   = #DIAG_BUF
                     057C  1232
                     057C  1233          $SETSFM_S ENBFLG = #1               ; Enable system service failure mode
                     0585  1234
                     0585  1235          $CANTIM_S REQIDT = #TIMEOUT1        ; OK, no timeout
                     0594  1236
41 0049'CF    07 E0  0594  1237          BBS      #LONGORSHRTV,FLAG,60$      ; Skip output if short report
                 52 D4  059A  1238          CLRL     R2                         ; Make an index
   53  0C2A'CF    DE  059C  1239          MOVAL    DIAG_BUF+HARD_ADR,R3       ; Set address address
   54  081C'CF    DE  05A1  1240          MOVAL    ARG_LIST,R4                ; Set arglist address
                     05A6  1241 50$:
       84  83  9A  05A6  1242          MOVZBL   (R3)+,(R4)+                ; Parse to bytes
     F9 52  06  F2  05A9  1243          AOBLSS   #6,R2,50$
                     05AD  1244
                     05AD  1245          $FAOL_S CTRSTR = ID_CTRSTR,-
                     05AD  1246                  OUTBUF = FAO_BUF,-
                     05AD  1247                  OUTLEN = BUFFER_PTR,-
                     05AD  1248                  PRMLST = ARG_LIST
                     05C4  1249
        0053'CF  DF  05C4  1250          PUSHAL   BUFFER_PTR
      00010001  8F  DD  05C8  1251          PUSHL    #^X10001
      00741133  8F  DD  05CE  1252          PUSHL    #UETP$_TEXT!STS$K_INFO
00000000'GF  03  FB  05D4  1253          CALLS    #3,G^LIB$SIGNAL
                     05DB  1254 60$:
    52  0C1A'CF  D0  05DB  1255          MOVL     DIAG_BUF+PCSR1,R2          ; Get the PCSR1 register contents
        03  52  B1  05E0  1256          CMPW     R2,#3                      ; Is it error free and ready (3)
            03  12  05E3  1257          BNEQ     70$                        ; BR if no
          0081  31  05E5  1258          BRW      100$                       ; BR if yes else...
                     05E8  1259 70$:
    18 52  0E  E1  05E8  1260          BBC      #PCSR1$V_ICAB,R2,80$       ; ...check the internal cable connection.
0769'CF  0BE6'CF  3C  05EC  1261          MOVZWL   XE_IOSB,STATUS             ; Set return status
        03AA'CF  DF  05F3  1262          PUSHAL   CAB2_UNPLUGED              ; Push error message address
            01  DD  05F7  1263          PUSHL    #1
      00741130  8F  DD  05F9  1264          PUSHL    #UETP$_TEXT!STS$K_WARNING ; Push message ID
            03  DD  05FF  1265          PUSHL    #3
          0F7B  31  0601  1266          BRW      ERROR_EXIT                 ; Untestable if internal cable unplugged
                     0604  1267 80$:
```

```
                2F 52    0F    E1  0604  1268          BBC     #PCSR1$V_XPWR,R2,90$     ; Check the external cable connection.
                   0765'CF    D6  0608  1269          INCL    ERROR_COUNT              ; Bump the error count
                   037D'CF    DF  060C  1270          PUSHAL  CAB1_UNPLUGED            ; Push error message address
               000F0001 8F    DD  0610  1271          PUSHL   #^XF0001
               00741130 8F    DD  0616  1272          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Push message ID
                   0765'CF    DD  061C  1273          PUSHL   ERROR_COUNT              ; Push the error count
                   06C3'CF    DF  0620  1274          PUSHAL  PROCESS_NAME
               00010002 8F    DD  0624  1275          PUSHL   #^X10002
               00748020 8F    DD  062A  1276          PUSHL   #UETP$_ERBOXPROC!STS$K_WARNING
          00000000'GF    07    FB  0630  1277          CALLS   #7,G^LIB$SIGNAL          ; Print the error message
                                0637  1278 90$:
                      08    EF  0637  1279          EXTZV   #PCSR1$V_SLFTST,-
                      06        0639  1280                  #PCSR1$S_SLFTST,-
                   53   52        063A  1281                  R2,R3                    ; Extract the self test error code
                                063C  1282
                                063C  1283          $FAO_S  CTRSTR = SLFTST_FAILED,-
                                063C  1284                  OUTBUF = FAO_BUF,-
                                063C  1285                  OUTLEN = BUFFER_PTR,-
                                063C  1286                  P1 = R3
                                0651  1287
          0769'CF    0BE6'CF    3C  0651  1288          MOVZWL  XE_IOSB,STATUS           ; Set return status
                   0053'CF    DF  0658  1289          PUSHAL  BUFFER_PTR               ; Push error message address
                      01      DD  065C  1290          PUSHL   #1
               00741132 8F    DD  065E  1291          PUSHL   #UETP$_TEXT!STS$K_ERROR  ; Push message ID
                      03      DD  0664  1292          PUSHL   #3
                    0F16      31  0666  1293          BRW     ERROR_EXIT               ; Mark as untestable
                                0669  1294 100$:
                    09EB      30  0669  1295          BSBW    CHECK_IOSB               ; Check the IOSB results and exit if bad
    5B  07B8'CF    000007B8'8F    C1  066C  1296          ADDL3   #UNIT_LIST,UNIT_LIST,R11 ; Get the unit block
                      02      88  0676  1297          BISB2   #UETUNT$M_TESTABLE,-
                   0B AB          0678  1298                  UETUNT$B_FLAGS(R11)      ; This one is good
                                067A  1299          $DASSGN_S CHAN = XE_CHAN1        ; Deassign the extra channel
                    08AE      31  0686  1300          BRW     CLEAN_EXIT               ; ONE SHOT is over
```

```
                                0689  1302 TIME_IT:
                                0689  1303
                                0689  1304 ;
                                0689  1305 ; Set up P1 device char buffer, P2 buffer is set up in Read/write section
                                0689  1306 ;
            53    0836'CF    DE  0689  1307          MOVAL   P1BUF+2,R3              ; Address of device char for p1
            83    05DC 8F    B0  068E  1308          MOVW    #MAX_UNABUF_SIZE,(R3)+  ; Maximum message length
                  63    02   90  0693  1309          MOVB    #XM$M_CHR_LOOPB,(R3)    ; Set loop back mode in char
              52    AA 8F    9A  0696  1310          MOVZBL  #^XAA,R2                ; Random number 1
                  53    2E   9A  069A  1311          MOVZBL  #^X2E,R3                ; Random number 2
         54    000005DC 8F    D0  069D  1312          MOVL    #MAX_UNABUF_SIZE,R4     ; Maximum message length
   5B 07B8'CF    000007B8'8F    C1  06A4  1313          ADDL3   #UNIT_LIST,UNIT_LIST,R11 ; Get the unit block address
            56    01A4 CB    DE  06AE  1314          MOVAL   UETUNT$K_DEVDEP(R11),R6 ; Get the transmit buffer address
            5A    0780 CB    DE  06B3  1315          MOVAL   UETUNT$K_DEVDEP+WRITE_SIZE(R11),R10 ; Get the receive buffer address
            58    01A4 CB    DE  06B8  1316          MOVAL   UETUNT$K_DEVDEP(R11),R8 ; Xmit address
         0BF8'CF    58    D0  06BD  1317          MOVL    R8,XMIT_BUF_ADR         ; Store xmit address
      0BFC'CF    05DC C8    DE  06C2  1318          MOVAL   WRITE_SIZE(R8),RECV_BUF_ADR ; Recv address
                                06C9  1319 10$:
                  52    53   C0  06C9  1320          ADDL2   R3,R2                   ; Random number as data
                  86    52   90  06CC  1321          MOVB    R2,(R6)+                ; Fill in the transmit buffer
                    F7  54   F5  06CF  1322          SOBGTR  R4,10$                  ; Branch if more bytes to be filled
                                06D2  1323
                                06D2  1324          $SETIMR_S DAYTIM = THREEMIN,-   ; Set timer AST to 3 minutes
                                06D2  1325                    ASTADR = TIME_SUC_OUT,-
                                06D2  1326                    EFN    = #EFN2
                                06E5  1327
                                06E5  1328          $TRNLOG_S LOGNAM = LOGNAM,-     ; Get the TESTNIADR, if any
                                06E5  1329                    RSLLEN = RSLLEN,-
                                06E5  1330                    RSLBUF = RSLBUF
                                06FE  1331
      00000000'8F    50    D1  06FE  1332          CMPL    R0,#SS$_NOTRAN          ; Was there any?
                  03    12  0705  1333          BNEQ    20$                     ; BR if not
                00A6    31  0707  1334          BRW     70$                     ; Otherwise forget it
                                070A  1335 20$:
      0C    07C8'CF    D1  070A  1336          CMPL    RSLLEN,#12              ; Was it the right length?
                  32    13  070F  1337          BEQL    30$                     ; BR if yes
            0765'CF    D6  0711  1338          INCL    ERROR_COUNT
            00F3'CF    DF  0715  1339          PUSHAL  NIADRWRONG              ; Report a bad address
         000F0001 8F    DD  0719  1340          PUSHL   #^XF0001
         00741130 8F    DD  071F  1341          PUSHL   #UETP$_TEXT!STS$K_WARNING
            0765'CF    DD  0725  1342          PUSHL   ERROR_COUNT
            06C3'CF    DF  0729  1343          PUSHAL  PROCESS_NAME
         00010002 8F    DD  072D  1344          PUSHL   #^X10002
         00748020 8F    DD  0733  1345          PUSHL   #UETP$_ERBOXPROC!STS$K_WARNING
   00000000'GF    07    FB  0739  1346          CALLS   #7,G^LIB$SIGNAL
                006D    31  0740  1347          BRW     70$
                                0743  1348 30$:
                  52    D4  0743  1349          CLRL    R2                      ; Make an index
      53    08F0'CF    DE  0745  1350          MOVAL   ADR2+6,R3               ; Set address address
      54    07D4'CF    DE  074A  1351          MOVAL   RSLBUF+8,R4             ; Set arglist address
                                074F  1352 40$:
            0814'CF    DF  074F  1353          PUSHAL  CVTRSL
                  64    9F  0753  1354          PUSHAB  (R4)
                  02    DD  0755  1355          PUSHL   #2
   00000000'GF    03    FB  0757  1356          CALLS   #3,G^LIB$CVT_HTB
      83    0814'CF    90  075E  1357          MOVB    CVTRSL,(R3)+
            54    02    C0  0763  1358          ADDL2   #2,R4
```

```
                E5 52    06   F2  0766  1359          AOBLSS  #6,R2,40$
             3D 0049'CF  07   E0  076A  1360          BBS     #LONGORSHRTV,FLAG,60$   ; Skip message if short report
                   52    D4   0770  1361          CLRL    R2                      ; Make an index
                53 08F0'CF  DE   0772  1362          MOVAL   ADR2+6,R3               ; Set address address
                54 081C'CF  DE   0777  1363          MOVAL   ARG_LIST,R4             ; Set arglist address
                           077C  1364 50$:
                   84 83    9A   077C  1365          MOVZBL  (R3)+,(R4)+             ; Parse to bytes
                F9 52    06   F2  077F  1366          AOBLSS  #6,R2,50$
                           0783  1367
                           0783  1368          $FAOL_S CTRSTR = NIADRTESTING,- ; Format address
                           0783  1369                  OUTBUF = FAO_BUF,-
                           0783  1370                  OUTLEN = BUFFER_PTR,-
                           0783  1371                  PRMLST = ARG_LIST
                           079A  1372
                   0053'CF  DF   079A  1373          PUSHAL  BUFFER_PTR              ; Output message
                         01  DD   079E  1374          PUSHL   #1
                00741133 8F  DD   07A0  1375          PUSHL   #UETP$_TEXT+STS$K_INFO
             00000000'GF  03  FB   07A6  1376          CALLS   #3,G^LIB$SIGNAL
                           07AD  1377 60$:
                      01A1  31   07AD  1378          BRW     XMIT_START
                           07B0  1379 70$:
                0862'CF  00   D0   07B0  1380          MOVL    #NMA$C_STATE_ON,PAD_SWITCH ; Set up dynamic characteristics
                0876'CF  00   D0   07B5  1381          MOVL    #NMA$C_STATE_ON,ECHO_SWITCH
          0868'CF  00000090 8F  D0   07BA  1382          MOVL    #LOOP_PROTOCOL,PROTOCOL
                           07C3  1383
                           07C3  1384          $SETIMR_S -                     ; Set up half minute timer
                           07C3  1385                  DAYTIM = HALFMIN,-      ; to prevent hang
                           07C3  1386                  ASTADR = TIME_ERR_OUT,-
                           07C3  1387                  REQIDT = #TIMEOUT2
                           07DA  1388 START_CONT:
                0818'CF  DF   07DA  1389          PUSHAL  XE_CHAN                 ; Start the UNA on the first channel
                0884'CF  DF   07DE  1390          PUSHAL  P2_DESC
             0FF8'CF  02  FB   07E2  1391          CALLS   #2,UNA_STARTUP
                           07E7  1392
                081A'CF  DF   07E7  1393          PUSHAL  XE_CHAN1                ; Start the UNA on the second channel
                08D4'CF  DF   07EB  1394          PUSHAL  P2_DESC1
             0FF8'CF  02  FB   07EF  1395          CALLS   #2,UNA_STARTUP
                           07F4  1396
             0049'CF  10  A8   07F4  1397          BISW2   #FLAG_SHUTDNM,FLAG      ; Set flag to say shut down the
                           07F9  1398                                          ; device if errors occur
                           07F9  1399
                           07F9  1400          $CANTIM_S REQIDT = #TIMEOUT2     ; Cancel hang timer
                           0808  1401
          08FC'CF  0C2A'CF  D0   0808  1402          MOVL    DIAG_BUF+HARD_ADR,SELF_ADR ; Set up NI return address
                0C2E'CF  B0   080F  1403          MOVW    DIAG_BUF+HARD_ADR+4,-
                0900'CF       0813  1404                  SELF_ADR+4
                           0816  1405
                           0816  1406          $SETIMR_S DAYTIM = TWOMIN,-      ; Set timer AST to 2 minutes
                           0816  1407                    ASTADR = REQUEST_NOTFND,-
                           0816  1408                    REQIDT = #TIME_ID_2
                           0829  1409
                           0829  1410          $QIO_S -                        ; Do a read of data
                           0829  1411                  CHAN   = XE_CHAN1,-
                           0829  1412                  FUNC   = #IO$_READVBLK,-
                           0829  1413                  EFN    = #RDID_EFN,-
                           0829  1414                  IOSB   = XE_IOSB,-
                           0829  1415                  ASTADR = REQUEST_FND,-
```

```
0829  1416                P1      = REQUEST_BUF,-
0829  1417                P2      = #REQUEST_BUFSIZ,-
0829  1418                P5      = #ADR2
085A  1419
085A  1420
```

```
                        085A  1422 RESTART:
                        085A  1423
                        085A  1424 ;
                        085A  1425 ; Loopback test transmit and receive random data with different message length
                        085A  1426 ;
                        085A  1427
                        085A  1428 XMIT_LOOP:
                        085A  1429          $SETIMR_S -                     ; Set half minute timer to prevent hang
                        085A  1430                  DAYTIM = HALFMIN,-
                        085A  1431                  ASTADR = TIME_ERR_OUT,-
                        085A  1432                  REQIDT = #TIMEOUT3
                        0871  1433
                     56 D4  0871  1434          CLRL    R6                      ; Set an index
                57 098E'CF DE  0873  1435          MOVAL   SIZE_TBL,R7             ; Get size table address
                        0878  1436 XMIT:
                        0878  1437          $QIO_S -                        ; Transmit data message
                        0878  1438                  EFN = #XMIT_EFN,-       ; Event flag
                        0878  1439                  CHAN = XE_CHAN,-        ; Channel
                        0878  1440                  FUNC = #IO$_WRITEVBLK,- ; Transmit
                        0878  1441                  IOSB = XE_IOSB,-        ; IOSB
                        0878  1442                  ASTADR = CHK_QIO_AST,-  ; Completion ast routine
                        0878  1443                  ASTPRM = #PRM,-         ; Ast parameter
                        0878  1444                  P1 = @XMIT_BUF_ADR,-    ; Addr of transmit buffer
                        0878  1445                  P2 = (R7)[R6],-         ; message length in bytes
                        0878  1446                  P5 = #DIAG_BUF+HARD_ADR ; NI address
                        08AA  1447
                        08AA  1448          $QIOW_S -                       ; Read data message
                        08AA  1449                  EFN = #RECV_EFN,-       ; Event flag
                        08AA  1450                  CHAN = XE_CHAN,-        ; Channel
                        08AA  1451                  FUNC = #IO$_READVBLK,-  ; Receive message
                        08AA  1452                  IOSB = RCV_IOSB,-       ; IOSB
                        08AA  1453                  ASTADR = RECV_AST,-     ; Completion ast to check data received
                        08AA  1454                  ASTPRM = (R7)[R6],-     ; Ast parameter = message length
                        08AA  1455                  P1 = @RECV_BUF_ADR,-    ; Receive buffer
                        08AA  1456                  P2 = (R7)[R6],-         ; Message length in bytes
                        08AA  1457                  P5 = #ADR1              ; NI address
                        08D9  1458
00 00000BFC'9F 00   2C  08D9  1459          MOVC5   #0,@#RECV_BUF_ADR,#0,-
    0BFC'DF 05DC 8F     08E1  1460                  #MAX_UNABUF_SIZE,@RECV_BUF_ADR ; Clear the read buffer
            0791'CF D6  08E7  1461          INCL    ITERATION               ; Increment iteration count
         89 56    04 F2  08EB  1462          AOBLSS  #SIZE_TBL_SIZE,R6,XMIT  ; Loop for size_tbl_size times
                        08EF  1463
                        08EF  1464          $CANTIM_S -                     ; Cancel hang timer
                        08EF  1465                  REQIDT = #TIMEOUT3
                        08FE  1466
   03 0049'CF    01 E1  08FE  1467          BBC     #TEST_OVERV,FLAG,10$    ; Is the test over?
                0435 31  0904  1468          BRW     SENSE_TEST
                        0907  1469 10$:
      0049'CF    06 E1  0907  1470          BBC     #BRDCST_NEDEDV,FLAG,-
                   03    090C  1471                  20$                     ; BR if UNA adr found
                0098 31  090D  1472          BRW      DO_BRDCST              ; BR  f no UNA adr found in 2 minute
                        0910  1473                                          ; Reac request
                        0910  1474 20$:
   03 0049'CF    05 E0  0910  1475          BBS     #ID_FNDV,FLAG,30$       ; BR if UNA adr found
                FF41 31  0916  1476          BRW     XMIT_LOOP               ; BR if no UNA adr found yet
                        0919  1477 30$:
                        0919  1478          $SETIMR_S -                     ; Set ten sec timer to prevent hang
```

```
                                0919  1479                DAYTIM = TENSEC,-
                                0919  1480                ASTADR = TIME_ERR_OUT,-
                                0919  1481                REQIDT = #TIMEOUT4
                                0930  1482
                     0818'CF    DF    0930  1483          PUSHAL  XE_CHAN                   ; Shutdown the UNA on the first channel
                102C'CF   01    FB    0934  1484          CALLS   #1,UNA_SHUTDOWN
                                0939  1485
                     081A'CF    DF    0939  1486          PUSHAL  XE_CHAN1                  ; Shutdown the UNA on the second channel
                102C'CF   01    FB    093D  1487          CALLS   #1,UNA_SHUTDOWN
                                0942  1488
                                0942  1489          $CANTIM_S REQIDT = #TIMEOUT4
                                0951  1490
                                0951  1491 XMIT_START:
                0862'CF   01    D0    0951  1492          MOVL    #NMA$C_STATE_OFF,PAD_SWITCH ; Turn off padding
                086E'CF   01    D0    0956  1493          MOVL    #NMA$C_STATE_OFF,ECHO_SWITCH ; Turn off echo mode
          0868'CF    00000090 8F    D0    095B  1494          MOVL    #LOOP_PROTOCOL,PROTOCOL ; Set to loop back protocol type
                                0964  1495
                                0964  1496          $SETIMR_S -                       ; Set twenty sec timer to prevent nang
                                0964  1497                  DAYTIM = TWENSEC,-
                                0964  1498                  ASTADR = TIME_ERR_OUT,-
                                0964  1499                  REQIDT = #TIMEOUT5
                                097B  1500
                     0818'CF    DF    097B  1501          PUSHAL  XE_CHAN                   ; Startup a UNA channel in loopback
                     0884'CF    DF    097F  1502          PUSHAL  P2_DESC
                OFF8'CF   02    FB    0983  1503          CALLS   #2,UNA_STARTUP
                                0988  1504
           08FC'CF   0C2A'CF    D0    0988  1505          MOVL    DIAG_BUF+HARD_ADR,SELF_ADR ; Set up NI return address
                     0C2E'CF    B0    098F  1506          MOVW    DIAG_BUF+HARD_ADR+4,-
                     0900'CF          0993  1507                  SELF_ADR+4
                                0996  1508
                                0996  1509
                                0996  1510          $CANTIM_S REQIDT = #TIMEOUT5
                                09A5  1511
                          0207  31    09A5  1512          BRW     XMIT_SETUP                ; Start external looping.
                                09A8  1513
                                09A8  1514 DO_BRDCST:
                                09A8  1515          $SETSFM_S ENBFLG = #0             ; Disable errors
                                09B1  1516          $CANCEL_S CHAN = XE_CHAN1         ; Give up the pending read request...
                                09BD  1517          $WAITFR_S EFN = #RDID_EFN         ; ...and wait for it to die
          0049'CF   0060 8F     AA    09C6  1518          BICW2   #<ID_FNDM!BRDCST_NEDEDM>,FLAG ; Init the ID found and brdcst flag
                                09CD  1519          $SETSFM_S ENBFLG = #1             ; Safe to turn on errors again
                                09D6  1520
                                09D6  1521          $SETIMR_S -                       ; Set ten sec timer to prevent hang
                                09D6  1522                  DAYTIM = TENSEC,-
                                09D6  1523                  ASTADR = TIME_ERR_OUT,-
                                09D6  1524                  REQIDT = #TIMEOUT4
                                09ED  1525
                     0818'CF    DF    09ED  1526          PUSHAL  XE_CHAN                   ; Shutdown the UNA on the first channel
                102C'CF   01    FB    09F1  1527          CALLS   #1,UNA_SHUTDOWN
                                09F6  1528
                     081A'CF    DF    09F6  1529          PUSHAL  XE_CHAN1                  ; Shutdown the UNA on the second channel
                102C'CF   01    FB    09FA  1530          CALLS   #1,UNA_SHUTDOWN
                                09FF  1531
                                09FF  1532          $CANTIM_S REQIDT = #TIMEOUT4
                                0A0E  1533
          08EA'CF    FFFFFFFF 8F    D0    0A0E  1534          MOVL    #-1,ADR2                  ; UNA address to broadcast
               08EE'CF    FFFF 8F    B0    0A17  1535          MOVW    #-1,ADR2+4
```

```
                0862'CF    01    D0  0A1E  1536          MOVL    #NMA$C_STATE_OFF,PAD_SWITCH ; Turn off padding
                086E'CF    01    D0  0A23  1537          MOVL    #NMA$C_STATE_OFF,ECHO_SWITCH ; Turn off echo mode
        C868'CF  00000090  8F    D0  0A28  1538          MOVL    #LOOP_PROTOCOL,PROTOCOL ; Set to loop back protocol type
                                     0A31  1539
                                     0A31  1540          $SETIMR_S -                     ; Set twenty sec timer to prevent hang
                                     0A31  1541                  DAYTIM = TWENSEC,-
                                     0A31  1542                  ASTADR = TIME_ERR_OUT,-
                                     0A31  1543                  REQIDT = #TIMEOUT5
                                     0A48  1544
                    0818'CF    DF    0A48  1545          PUSHAL  XE_CHAN                 ; Startup a UNA channel in loopback
                    0884'CF    DF    0A4C  1546          PUSHAL  P2_DESC
                OFF8'CF    02    FB  0A50  1547          CALLS   #2,UNA_STARTUP
                                     0A55  1548
                                     0A55  1549          $CANTIM_S REQIDT = #TIMEOUT5
                                     0A64  1550
                                     0A64  1551          $SETSFM_S ENBFLG = #0           ; If not don't error hard
                                     0A6D  1552
                                     0A6D  1553          $SETIMR_S -                     ; Set ten sec timer to prevent hang
                                     0A6D  1554                  DAYTIM = TENSEC,-
                                     0A6D  1555                  ASTADR = TIME_ERR_OUT,-
                                     0A6D  1556                  REQIDT = #TIMEOUT6
                                     0A84  1557
                                     0A84  1558          $QIO_S CHAN = XE_CHAN,-         ; Xmit to the broadcast address
                                     0A84  1559                  FUNC = #IO$_WRITEVBLK,-
                                     0A84  1560                  IOSB = XE_IOSB,-
                                     0A84  1561                  ASTADR = CHK_QIO_AST,-  ; Completion ast routine
                                     0A84  1562                  ASTPRM = #PRM,-         ; Ast parameter
                                     0A84  1563                  P1 = BRDCST_BUF,-       ; Addr of transmit buffer
                                     0A84  1564                  P2 = #BRDCST_BUF_LEN,-  ; message length in bytes
                                     0A84  1565                  P5 = #ADR2              ; NI broadcast address
                                     0AB5  1566
                                     0AB5  1567          $CANTIM_S REQIDT = #TIMEOUT6
                                     0AC4  1568
                                     0AC4  1569          $SETIMR_S DAYTIM = TENSEC,-     ; Should get a response in ten seconds
                                     0AC4  1570                  ASTADR = ABRT_BRDCST,-
                                     0AC4  1571                  REQIDT = #TIME_ID_2
                                     0AD7  1572
                                     0AD7  1573          $QIOW_S CHAN = XE_CHAN,-        ; Receive from the broadcast address
                                     0AD7  1574                  FUNC = #IO$_READVBLK,-
                                     0AD7  1575                  IOSB = XE_IOSB,-
                                     0AD7  1576                  ASTADR = REQUEST_FND,-  ; Completion ast routine
                                     0AD7  1577                  ASTPRM = #PRM,-         ; Ast parameter
                                     0AD7  1578                  P1 = @RECV_BUF_ADR,-    ; Addr of receive buffer
                                     0AD7  1579                  P2 = #MAX_ONABUF_SIZE,- ; message length in bytes
                                     0AD7  1580                  P5 = #ADR2              ; NI address
                                     0B0C  1581
    0BF6'CF    0BE6'CF    B0    0B0C  1582          MOVW    XE_IOSB,SAVE_STAT       ; Save the read status
                                     0B13  1583
                                     0B13  1584          $SETIMR_S -                     ; Set ten sec timer to prevent hang
                                     0B13  1585                  DAYTIM = TENSEC,-
                                     0B13  1586                  ASTADR = TIME_ERR_OUT,-
                                     0B13  1587                  REQIDT = #TIMEOUT7
                                     0B2A  1588
                    0818'CF    DF    0B2A  1589          PUSHAL  XE_CHAN                 ; Shutdown the UNA
                102C'CF    01    FB  0B2E  1590          CALLS   #1,UNA_SHUTDOWN
                                     0B33  1591
                                     0B33  1592          $CANTIM_S REQIDT = #TIMEOUT7
```

```
                     0B42  1593
                0A 0BF6'CF    E8  0B42  1594          BLBS    SAVE_STAT,10$              ; BR if read made it
          0862'CF    00       D0  0B47  1595          MOVL    #NMA$C_STATE_ON,PAD_SWITCH ; Turn padding back on
          086E'CF    00       D0  0B4C  1596          MOVL    #NMA$C_STATE_ON,ECHO_SWITCH ; Turn echo back on
                     0B51  1597 10$:
                     0B51  1598
                     0B51  1599          $SETIMR_S -                        ; Set twenty sec timer to prevent hang
                     0B51  1600                  DAYTIM = TWENSEC,-
                     0B51  1601                  ASTADR = TIME_ERR_OUT,-
                     0B51  1602                  REQIDT = #TIMEOUT8
                     0B68  1603
              0818'CF    DF  0B68  1604          PUSHAL  XE_CHAN                    ; Restart the UNA
              0884'CF    DF  0B6C  1605          PUSHAL  P2_DESC
          0FF8'CF    02       FB  0B70  1606          CALLS   #2,UNA_STARTUP
                     0B75  1607
                     0B75  1608          $CANTIM_S REQIDT = #TIMEOUT8
                     0B84  1609
                26 0BF6'CF    E8  0B84  1610          BLBS    SAVE_STAT,XMIT_SETUP       ; If we got one then start external loopback
                     0B89  1611
              0200'CF    DF  0B89  1612          PUSHAL  REMOTE_NOTFND              ; Else report an error
                 01      DD  0B8D  1613          PUSHL   #1
           00741130  8F      DD  0B8F  1614          PUSHL   #UETP$_TEXT!STS$K_WARNING
     00000000'GF     03       FB  0B95  1615          CALLS   #3,G^LIB$SIGNAL
                     0B9C  1616
                     0B9C  1617          $SETSFM_S ENBFLG = #1              ; It's OK now
    0049'CF    0040  8F      AA  0BA5  1618          BICW2   #BRDCST_NEDEDM,FLAG        ; We're giving up
                FCAB      31  0BAC  1619          BRW     XMIT_LOOP                  ; Carry on in Echo mode
```

```
                                0BAF  1621 ;
                                0BAF  1622 ; When requesting looped back messages from a remote node, we have to account
                                0BAF  1623 ; for the Ethernet's uncertainty:  messages are not guaranteed to reach their
                                0BAF  1624 ; destination and it is the responsibility of the sender to retransmit them
                                0BAF  1625 ; if they do not.  At the same time, UETP must terminate.  We therefore limit
                                0BAF  1626 ; the number of attempts we will make at retransmission and the time we will
                                0BAF  1627 ; wait for any one attempt to transmit.  Any I/O is $CANCELled after it
                                0BAF  1628 ; completes or after we time out; if it were to complete later in the test it
                                0BAF  1629 ; could give us inconsistent results.
                                0BAF  1630 ;
                                0BAF  1631
                                0BAF  1632 XMIT_SETUP:
                         0E   28 0BAF  1633         MOVC3   #FORWARD_MSG_SIZ,-          ; Put the forward message header in...
          0BF8'DF        0BF8'CF 0BB1  1634                 BRDCST_BUF,@XMIT_BUF_ADR ; ...the transmit buffer
0100 8F    00    00 8F    00   2C 0BB7  1635         MOVC5   #0,#0,#0,#TSCLL,-           ; Clear out time stamps from previous...
                   0C5C'CF        0BBF  1636                 STAMP_BUFFER             ; ...run (needed for LOOP mode)
                                0BC2  1637
                                0BC2  1638 REMOTE_LOOPBACK_TEST:
      04    00    00 8F    00   2C 0BC2  1639         MOVC5   #0,#0,#0,#SIZE_TBL_SIZE,- ; Clear flags which say that some...
                   0C54'CF        0BC8  1640                 LOOPBACK_ACK             ; ...transmission size succeeded
                         56    7C 0BCB  1641         CLRQ    R6                       ; Initialize counter for msg sizes...
                                0BCD  1642                                          ; ...and retransmission flag
                58   099E'CF   DE 0BCD  1643         MOVAL   SIZE_TBL1,R8             ; Point to table of message sizes
                59   0818'CF   3C 0BD2  1644         MOVZWL  XE_CHAN,R9               ; Stick channel in convenient holder
                                0BD7  1645         $SETIMR_S DAYTIM = HALFMIN,-     ; Allow time for messages to complete
                                0BD7  1646                   ASTADR = LOOPBACK_TMO,-
                                0BD7  1647                   REQIDT = #LOOPBACK_TMO
                   0791'CF   D6 0BEE  1648 10$:     INCL    ITERATION                ; Count the messages we're to receive
                         5A    D4 0BF2  1649         CLRL    R10                      ; Limits retransmission attempts
05DC 8F    00    00 8F    00   2C 0BF4  1650         MOVC5   #0,#0,#0,#MAX_UNABUF_SIZE,- ; Clear buffer into which...
                   0BFC'DF        0BFC  1651                 @RECV_BUF_ADR            ; ...messages are read
                                0BFF  1652         $QIO_S  EFN    = #RECV_EFN,-     ; Be ready to read if the msg arrives
                                0BFF  1653                 CHAN   = R9,-
                                0BFF  1654                 FUNC   = #IO$_READLBLK,-
                                0BFF  1655                 IOSB   = RCV_IOSB,-
                                0BFF  1656                 ASTADR = LOOPBACK_AST,-
                                0BFF  1657                 ASTPRM = R6,-
                                0BFF  1658                 P1     = @RECV_BUF_ADR,-
                                0BFF  1659                 P2     = (R8),-
                                0BFF  1660                 P5     = #ADR1           ; NI address
                                0C2A  1661
          0B 0049'CF    E9 0C2A  1662         BLBC    FLAG,20$                     ; Skip sequence number if normal running
       5B    0BF8'CF    D0 0C2F  1663         MOVL    XMIT_BUF_ADR,R11             ; In dump mode, use sequence number...
0A AB    0791'CF    D0 0C34  1664         MOVL    ITERATION,10(R11)            ; ...(ITERATION) to verify message
             0A AB    DD 0C3A  1665 20$:     PUSHL   10(R11)
                68    DD 0C3D  1666         PUSHL   (R8)
   121B'CF    02    FB 0C3F  1667         CALLS   #2,XMIT_STAMP                ; In dump mode, keep track of messages
                                0C44  1668         $QIOW_S EFN    = #XMIT_EFN,-     ; Tell some node to loop back a message
                                0C44  1669                 CHAN   = R9,-
                                0C44  1670                 FUNC   = #IO$_WRITELBLK,-
                                0C44  1671                 IOSB   = XE_IOSB,-
                                0C44  1672                 P1     = @XMIT_BUF_ADR,-
                                0C44  1673                 P2     = (R8),-
                                0C44  1674                 P5     = #ADR2+6         ; Remote NI address
                                0C6B  1675         $WAITFR_S EFN = #RECV_EFN        ; Wait for message to be returned
       38 0C54'C6    E8 0C74  1676         BLBS    LOOPBACK_ACK(R6),30$         ; BR if this size message got through
          57    01    D0 0C79  1677         MOVL    #1,R7                        ; Indicate retransmission
```

```
                        0C7C  1678          $CLREF_S EFN = #RECV_EFN         ; Flag was set by timeout routine
                5A   D6 0C85  1679          INCL    R10                      ; Figure which retransmission is next
      20 0049'CF    E9 0C87  1680          BLBC    FLAG,25$                 ; Skip message if normal running
                        0C8C  1681          $FAO_S  CTRSTR = RETRANS_MSG,-
                        0C8C  1682                  OUTLEN = BUFFER_PTR,-
                        0C8C  1683                  OUTBUF = FAO_BUF,-
                        0C8C  1684                  P1     = R10,-
                        0C8C  1685                  P2     = ITERATION,-
                        0C8C  1686                  P3     = (R8),-
                        0C8C  1687                  P4     = #0
                0611    30 0CA9  1688          BSBW    DUMP_SIGNAL
             5A   04    D1 0CAC  1689 25$:     CMPL    #RETRY_COUNT,R10         ; Loop to re-xmit...
                  89    14 0CAF  1690          BGTR    20$                      ; ...if any attempts left
                  88    D5 0CB1  1691 30$:     TSTL    (R8)+                    ; Point to the next message size
FF35 56   01    03    F1 0CB3  1692          ACBL    #SIZE_TBL_SIZE-1,#1,R6,10$ ; Loop while there are more msg sizes
            46  57    E9 0CB9  1693          BLBC    R7,70$                   ; Skip msg if no retransmissions
             5A 5E    D0 0CBC  1694          MOVL    SP,R10                   ; Save original SP for later cleanup
                56    7C 0CBF  1695          CLRQ    R6                       ; Counter for msg sizes which failed...
                        0CC1  1696                                           ; ...(R7) and loop index (R6)
          07 0C54'C6    E8 0CC1  1697 40$:     BLBS    LOOPBACK_ACK(R6),50$     ; BR if this size message got across
                57    D6 0CC6  1698          INCL    R7                       ; Count size(s) which failed
           099E'CF46    DD 0CC8  1699          PUSHL   SIZE_TBL1[R6]            ; Save the message size
             F0 56  04    F2 0CCD  1700 50$:     AOBLSS  #SIZE_TBL_SIZE,R6,40$    ; Loop through remainder of flags
          58  05E9'CF    DE 0CD1  1701          MOVAL   RETRY_MSG,R8             ; Assume we want msg for all sizes OK
                04    DD 0CD6  1702          PUSHL   #SIZE_TBL_SIZE           ; Save total count of message sizes
                57    D5 0CD8  1703          TSTL    R7                       ; Did all msg sizes eventually make it?
                0A    13 0CDA  1704          BEQL    60$                      ; BR if they did
          58  0667'CF    DE 0CDC  1705          MOVAL   LOOPBACK_FAIL_MSG,R8     ; Use other message if they did not
                6E  57    D0 0CE1  1706          MOVL    R7,(SP)                  ; Finally, save count of failed msgs...
                57    DD 0CE4  1707          PUSHL   R7                       ; ...for !%S and !# both
                59  5E    D0 0CE6  1708 60$:     MOVL    SP,R9                    ; Save pointer to $FAO args
                        0CE9  1709          $FAOL_S CTRSTR = (R8),-          ; Form msg indicating fails or attempts
                        0CE9  1710                  OUTBUF = FAO_BUF,-
                        0CE9  1711                  OUTLEN = BUFFER_PTR,-
                        0CE9  1712                  PRMLST = (R9)
                05BE    30 0CFC  1713          BSBW    DUMP_SIGNAL              ; Tell the user of our difficulties
             5E   5A    D0 0CFF  1714          MOVL    R10,SP                   ; Clean $FAOL args from the stack
                        0D02  1715 70$:     $CANCEL_S CHAN = XE_CHAN         ; We can't afford to get old msgs later
                        0D0E  1716          $CANTIM_S REQIDT = #LOOPBACK_TMO ; Now we know test won't hang
                        0D1D  1717
19 0049'CF    01    E0 0D1D  1718          BBS     #TEST_OVERV,FLAG,SENSE_TEST ; Is the test over?
              FE9C  31 0D23  1719          BRW     REMOTE_LOOPBACK_TEST     ; Continue in external looping
```

```
                                0D26  1721 ;
                                0D26  1722 ; If in ten seconds the transmit to the broadcast address does not succeed
                                0D26  1723 ; either the device is broken or there is no one else on the NI so abort
                                0D26  1724 ; the read request and continue looping in Echo mode.
                                0D26  1725 ;
                                0D26  1726
                                0D26  1727 ABRT_BRDCST:
                          0000  0D26  1728         .WORD   0
      0049'CF    0040 8F    A8  0D28  1729         BISW2   #BRDCST_NEDEDM,FLAG     ; Signal the CANCELed AST routine
                                0D2F  1730         $CANCEL_S CHAN = XE_CHAN        ; Give up completely
                             04  0D3B  1731         RET
                                0D3C  1732
                                0D3C  1733 SENSE_TEST:
                                0D3C  1734
                                0D3C  1735         $SETIMR_S -                     ; Set ten sec timer to prevent hang
                                0D3C  1736                 DAYTIM = TENSEC,-
                                0D3C  1737                 ASTADR = TIME_ERR_OUT,-
                                0D3C  1738                 REQIDT = #TIMEOUT10
                                0D53  1739
                                0D53  1740         $QIOW_S -                       ; Read device characteristics
                                0D53  1741                 CHAN = XE_CHAN,-
                                0D53  1742                 FUNC = #IO$_SENSEMODE!IO$M_CTRL,-
                                0D53  1743                 IOSB = XE_IOSB,-
                                0D53  1744                 P1 = SENSE_P1BUF,-
                                0D53  1745                 P2 = #SENSE_P2DESC
                                0D7A  1746
                                0D7A  1747         $CANTIM_S REQIDT = #TIMEOUT10
                                0D89  1748
                   02CB     30  0D89  1749         BSBW    CHECK_IOSB              ; Check status
                                0D8C  1750
            54  0BE8'CF    3C  0D8C  1751         MOVZWL  XE_IOSB+2,R4            ; Number of bytes returned for p2 buff
            55  087E'CF    DE  0D91  1752         MOVAL   ADR,R5                  ; Digital multicast NI address
                57    06    D0  0D96  1753         MOVL    #6,R7                   ; Adr length
                                0D99  1754 10$:
            56  0936'CF    DE  0D99  1755         MOVAL   SENSE_P2BUF,R6          ; Address of P2 buff returned
      66    54  65    06    39  0D9E  1756         MATCHC  #6,(R5),R4,(R6)         ; Check the parameters returned
                      03    12  0DA3  1757         BNEQ    30$                     ; Br if not match
                    0007    31  0DA5  1758         BRW     ERROR_TEST              ; Otherwise go to test error case
                                0DA8  1759 30$:
                0A3A'CF    DF  0DA8  1760         PUSHAL  SENSE_ERRMSG            ; Error message
                    0237    31  0DAC  1761         BRW     FAIL_OUT                ; Failure exit
                                0DAF  1762
                                0DAF  1763 ERROR_TEST:
                                0DAF  1764         $SETSFM_S ENBFLG = #0           ; Turn off system service mode
                                0DB8  1765
                                0DB8  1766 ;
                                0DB8  1767 ; Read data with IO$M_NOW specified but no data available
                                0DB8  1768 ;
                                0DB8  1769         $QIOW_S -                       ; Read data message
                                0DB8  1770                 CHAN = XE_CHAN,-
                                0DB8  1771                 FUNC = #IO$_READVBLK!IO$M_NOW,-
                                0DB8  1772                 IOSB = XE_IOSB,-
                                0DB8  1773                 P1 = @RECV_BUF_ADR,-
                                0DB8  1774                 P2 = #MAX_ONABOF_SIZE
                                0DDF  1775
      0000'8F   0BE6'CF    B1  0DDF  1776         CMPW    XE_IOSB,#SS$_ENDOFFILE  ; Correct error code?
                      69    12  0DE6  1777         BNEQ    ERRTST_ERR              ; Br if not
```

```
                        0DE8  1778
                        0DE8  1779 ;
                        0DE8  1780 ; Buffer not enough to hold all information from IO$_SENSEMODE
                        0DE8  1781 ;
                        0DE8  1782
                        0DE8  1783          $QIOW_S -                       ; Read device characteristics
                        0DE8  1784                  CHAN = XE_CHAN,-
                        0DE8  1785                  FUNC = #IO$_SENSEMODE!IO$M_CTRL,-
                        0DE8  1786                  IOSB = XE_IOSB,-
                        0DE8  1787                  P1 = SENSE_P1BUF,-
                        0DE8  1788                  P2 = #ERRTST_P2DESC
                        0E0F  1789
0000'8F  0BE6'CF  B1    0E0F  1790          CMPW    XE_IOSB,#SS$_BUFFEROVF  ; Error code = buffer overflow?
              39  12    0E16  1791          BNEQ    ERRTST_ERR              ; Error if not
                        0E18  1792
                        0E18  1793 ;
                        0E18  1794 ; Bad parameter due to message too big for the buffer
                        0E18  1795 ;
                        0E18  1796          $QIOW_S -                       ; Transmit data message with leng = 514
                        0E18  1797                  EFN = #XMIT_EFN,-
                        0E18  1798                  CHAN = XE_CHAN,-
                        0E18  1799                  FUNC = #IO$_WRITEVBLK,-
                        0E18  1800                  IOSB = XE_IOSB,-
                        0E18  1801                  P1 = @XMIT_BUF_ADR,-
                        0E18  1802                  P2 = #MAX_ONABOF_SIZE+2,-
                        0E18  1803                  P5 = #DIAG_BUF+HARD_ADR
                        0E45  1804
00000000'8F    50   D1    0E45  1805          CMPL    R0,#SS$_IVBUFLEN        ; Should be invalid buffer size
                03  12    0E4C  1806          BNEQ    ERRTST_ERR
              0076  31    0E4E  1807          BRW     READ_ERRCOUNT           ; Br to read and clear error count
                        0E51  1808
                        0E51  1809 ERRTST_ERR:
        0A5B'CF   DF    0E51  1810          PUSHAL  ERRTEST_MSG             ; Error message
           018E   31    0E55  1811          BRW     FAIL_OUT                ; Failure exit
                        0E58  1812
                        0E58  1813 REQUEST_END:
                0FFC    0E58  1814          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
5C 0049'CF      06  E0  0E5A  1815          BBS     #BRDCST_NEDEDV,FLAG,20$ ; Skip this if being CANCELed
3D 0049'CF      07  E0  0E60  1816          BBS     #LONGORSHRTV,FLAG,15$   ; Skip message if short report
              52  D4    0E66  1817          CLRL    R2                      ; Make an index
   53   08F0'CF  DE    0E68  1818          MOVAL   ADR2+6,R3               ; Set address address
   54   081C'CF  DE    0E6D  1819          MOVAL   ARG_LIST,R4             ; Set arglist address
                        0E72  1820 10$:
        84   83  9A    0E72  1821          MOVZBL  (R3)+,(R4)+             ; Parse to bytes
     F9 52   06  F2    0E75  1822          AOBLSS  #6,R2,10$
                        0E79  1823
                        0E79  1824          $FAOL_S CTRSTR = REMOTE_ID_CTR,- ; Format address
                        0E79  1825                  OUTBUF = FAO_BUF,-
                        0E79  1826                  OUTLEN = BUFFER_PTR,-
                        0E79  1827                  PRMLST = ARG_LIST
                        0E90  1828
        0053'CF   DF    0E90  1829          PUSHAL  BUFFER_PTR              ; Output message
              01  DD    0E94  1830          PUSHL   #1
     00741133   8F  DD  0E96  1831          PUSHL   #UETP$_TEXT+STS$K_INFO
00000000'GF    03  FB  0E9C  1832          CALLS   #3,G^LIB$SIGNAL
                        0EA3  1833 15$:
                        0EA3  1834          $CANTIM_S REQIDT = #TIME_ID_2    ; Kill two minute timer
```

```
                        0EAE  1835
                 081A'CF DF   0EAE  1836          PUSHAL  XE_CHAN1                ; Shutdown the read request channel
           102C'CF    01 FB   0EB2  1837          CALLS   #1,UNA_SHUTDOWN
                        0EB7  1838
           0049'CF    20 A8   0EB7  1839          BISW2   #ID_FNDM,FLAG           ; Set request found
                        0EBC  1840 20$:
                        04    0EBC  1841          RET
                        0EBD  1842
                        0EBD  1843 REQUEST_NOTFND:
                      0FFC    0EBD  1844          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        0EBF  1845
     0049'CF    0040 8F A8   0EBF  1846          BISW2   #BRDCST_NEDEDM,FLAG     ; Set the broadcast needed flag
                        04    0EC6  1847          RET
                        0EC7  1848
                        0EC7  1849 READ_ERRCOUNT:
                        0EC7  1850          $SETSFM_S ENBFLG = #1           ; Turn on  system service mode
                        0ED0  1851
                        0ED0  1852          $SETIMR_S -                     ; Set ten sec timer to prevent hang
                        0ED0  1853                  DAYTIM = TENSEC,-
                        0ED0  1854                  ASTADR = TIME_ERR_OUT,-
                        0ED0  1855                  REQIDT = #TIMEOUT1
                        0EE7  1856
                        0EE7  1857          $QIOW_S -                       ; Read and clear the error counters
                        0EE7  1858                  CHAN = XE_CHAN,-
                        0EE7  1859                  FUNC = #IO$_SENSEMODE!IO$M_RD_COUNT!IO$M_CLR_COUNT!IO$M_CTRL,-
                        0EE7  1860                  IOSB = XE_IOSB,-
                        0EE7  1861                  P2 = #ERRCOUNT_DESC
                        0F0C  1862
                        0F0C  1863          $CANTIM_S REQIDT = #TIMEOUT11
                        0F1B  1864
                  0139  30    0F1B  1865          BSBW    CHECK_IOSB              ; Check IO status block
     09D6'CF    0BE8'CF 3C    0F1E  1866          MOVZWL  XE_IOSB+2,ERRCOUNT_DESC ; Set the counter size
                09D6'CF DF    0F25  1867          PUSHAL  ERRCOUNT_DESC           ; Push address of counters
           12D1'CF    01 FB   0F29  1868          CALLS   #1,COUNTER_CHECK        ; and check the counters for errors
09D6'CF     00000080 8F D0    0F2E  1869          MOVL    #ERRCNT_LEN,ERRCOUNT_DESC ; Reset the buffer size
                        0F37  1870
                        0F37  1871 CLEAN_EXIT:
                        0F37  1872
           0049'CF    10 AA   0F37  1873          BICW2   #FLAG_SHUTDNM,FLAG      ; Clear the shutdown flag
                        0F3C  1874
                        0F3C  1875          $QIOW_S -                       ; Shut down the device
                        0F3C  1876                  CHAN = XE_CHAN,-
                        0F3C  1877                  FUNC = #IO$_SETMODE!IO$M_CTRL!IO$M_SHUTDOWN,-
                        0F3C  1878                  IOSB = XE_IOSB
                        0F5D  1879
                  00F7  30    0F5D  1880          BSBW    CHECK_IOSB              ; Check IO status b.ock
                        0F60  1881
                        0F60  1882 SUC_EXIT:
                        0F60  1883          $TRNLOG_S LOGNAM = MODE,-
                        0F60  1884                    RSLLEN = BUFFER_PTR,-
                        0F60  1885                    RSLBUF = FAO_BUF      ; Get the run mode
        005B'CF    20 8A    0F79  1886          BICB2   #LC_BITM,BUFFER         ; Convert to upper case
     005B'CF    4C 8F 91    0F7E  1887          CMPB    #^A/L/,BUFFER           ; Is this a loop for ever?
                40 12       0F84  1888          BNEQ    10$                     ; BR if not
           0049'CF    22 AA   0F86  1889          BICW2   #<TEST_OVERM!ID_FNDM>,FLAG ; Reset the termination flag
                0795'CF D6    0F8B  1890          INCL    PASS                    ; Bump the pass count
                        0F8F  1891          $FAO_S  CTRSTR = PASS_MSG,-
```

```
                                0F8F  1892                OUTLEN = BUFFER_PTR,-
                                0F8F  1893                OUTBUF = FAO_BUF,-
                                0F8F  1894                P1     = PASS,-
                                0F8F  1895                P2     = ITERATION,-
                                0F8F  1896                P3     = #0                    ; Make the end of pass message
                0053'CF    DF   0FAC  1897          PUSHAL  BUFFER_PTR                  ; Push the string desc.
                      01   DD   0FB0  1898          PUSHL   #1                          ; Push arg count
                00741133   8F   DD   0FB2  1899          PUSHL   #UETP$_TEXT!STS$K_INFO      ; Push the signal name
            00000000'GF    03   FB   0FB8  1900          CALLS   #3,G^LIB$SIGNAL             ; Print the end of pass message
                0791'CF    D4   0FBF  1901          CLRL    ITERATION                   ; Reset the iteration count
                   F6C3    31   0FC3  1902          BRW     TIME_IT                     ; Do the next pass
                                0FC6  1903 10$:
                                0FC6  1904          $DASSGN_S CHAN = XE_CHAN            ; All done with the channel
                                0FD2  1905
    0769'CF    10000000'8F    D0   0FD2  1906          MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,STATUS ; Set successful exit status
                                0FDB  1907          $EXIT_S STATUS                      ; Exit with the status
                                0FE6  1908
                                0FE6  1909 FAIL_OUT:                                    ; Failure exit
                      01   DD   0FE6  1910          PUSHL   #1                          ; Arg count
                00741132   8F   DD   0FE8  1911          PUSHL   #UETP$_TEXT!STS$K_ERROR     ; Signal name
                0769'CF    6E   D0   0FEE  1912          MOVL    (SP),STATUS                 ; Set up status
                      03   DD   0FF3  1913          PUSHL   #3                          ; Arg count
                    0587   31   0FF5  1914          BRW     ERROR_EXIT                  ; Error exit
                                0FF8  1915
```

```
                     0FF8  1917          .SBTTL  UNA Startup Routine
                     0FF8  1918 ;++
                     0FF8  1919 ; FUNCTIONAL DESCRIPTION:
                     0FF8  1920 ;       This routine will be called to start up the UNA device.
                     0FF8  1921 ;       It checks IO status block and the AST parameter
                     0FF8  1922 ;
                     0FF8  1923 ; CALLING SEQUENCE:
                     0FF8  1924 ;       PUSHAL CHAN
                     0FF8  1925 ;       PUSHAL P2
                     0FF8  1926 ;       CALLS   #2,UNA_STARTUP
                     0FF8  1927 ;
                     0FF8  1928 ; INPUT PARAMETERS:
                     0FF8  1929 ;       CHAN = Address to store the returned channel in
                     0FF8  1930 ;       P2   = Address of the P2 parameter list for the UNA
                     0FF8  1931 ;
                     0FF8  1932 ; IMPLICIT INPUTS:
                     0FF8  1933 ;       NONE
                     0FF8  1934 ;
                     0FF8  1935 ; OUTPUT PARAMETERS:
                     0FF8  1936 ;       CHAN = channel associated to the UNA device on startup
                     0FF8  1937 ;
                     0FF8  1938 ; IMPLICIT OUTPUTS:
                     0FF8  1939 ;       Error message if error
                     0FF8  1940 ;
                     0FF8  1941 ; COMPLETION CODES:
                     0FF8  1942 ;       IO status in XE_IOSB if error
                     0FF8  1943 ;
                     0FF8  1944 ; SIDE EFFECTS:
                     0FF8  1945 ;       Program exit if error
                     0FF8  1946 ;
                     0FF8  1947 ;--
                     0FF8  1948
                     0FF8  1949 UNA_STARTUP:
            0FFC     0FF8  1950          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0FFA  1951
                     0FFA  1952          $QIOW_S -                       ; Start the controller
                     0FFA  1953                  CHAN = @8(AP),-
                     0FFA  1954                  FUNC = #IO$_SETCHAR!IO$M_CTRL!IO$M_STARTUP,-
                     0FFA  1955                  IOSB = XE_IOSB,-
                     0FFA  1956                  ASTADR = CHK_QIO_AST,-
                     0FFA  1957                  ASTPRM = #PRM,-
                     0FFA  1958                  P2 = 4(AP),-
                     0FFA  1959                  P3 = #RECVPOOL_SIZ,-
                     0FFA  1960                  P6 = #DIAG_BUF
              04     102B  1961          RET
```

```
                     102C  1963          .SBTTL  UNA Shutdown Routine
                     102C  1964 ;++
                     102C  1965 ; FUNCTIONAL DESCRIPTION:
                     102C  1966 ;       This routine will be called to shutdown the UNA device.
                     102C  1967 ;       It checks IO status block and the AST parameter
                     102C  1968 ;
                     102C  1969 ; CALLING SEQUENCE:
                     102C  1970 ;       PUSHAL CHAN
                     102C  1971 ;       CALLS   #1,UNA_SHUTDOWN
                     102C  1972 ;
                     102C  1973 ; INPUT PARAMETERS:
                     102C  1974 ;       CHAN = Address to store the returned channel in
                     102C  1975 ;
                     102C  1976 ; IMPLICIT INPUTS:
                     102C  1977 ;       NONE
                     102C  1978 ;
                     102C  1979 ; OUTPUT PARAMETERS:
                     102C  1980 ;       NONE
                     102C  1981 ;
                     102C  1982 ; IMPLICIT OUTPUTS:
                     102C  1983 ;       Error message if error
                     102C  1984 ;
                     102C  1985 ; COMPLETION CODES:
                     102C  1986 ;       IO status in XE_IOSB if error
                     102C  1987 ;
                     102C  1988 ; SIDE EFFECTS:
                     102C  1989 ;       Program exit if error
                     102C  1990 ;
                     102C  1991 ;--
                     102C  1992
                     102C  1993 UNA_SHUTDOWN:
                OFFC 102C  1994          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     102E  1995
                     102E  1996          $QIOW_S -                       ; Shut down the device
                     102E  1997                  CHAN = @4(AP),-
                     102E  1998                  FUNC = #IO$_SETMODE!IO$M_CTRL!IO$M_SHUTDOWN,-
                     102E  1999                  IOSB = XE_IOSB,-
                     102E  2000                  ASTADR = CHK_QIO_AST,-
                     102E  2001                  ASTPRM = #PRM
                  04 1056  2002          RET
```

```
                     1057  2004          .SBTTL  CHECKIOSB - Check IO status block
                     1057  2005 ;++
                     1057  2006 ; FUNCTIONAL DESCRIPTION:
                     1057  2007 ;       This routine checks the IO status block = #SS$_NORMAL
                     1057  2008 ;
                     1057  2009 ; CALLING SEQUENCE:
                     1057  2010 ;       BSBW    CHECK_IOSB
                     1057  2011 ;
                     1057  2012 ; INPUT PARAMETERS:
                     1057  2013 ;       NONE
                     1057  2014 ;
                     1057  2015 ; IMPLICIT INPUTS:
                     1057  2016 ;       NONE
                     1057  2017 ;
                     1057  2018 ; OUTPUT PARAMETERS:
                     1057  2019 ;       NONE
                     1057  2020 ;
                     1057  2021 ; IMPLICIT OUTPUTS:
                     1057  2022 ;       Exit with status if IOSB not right
                     1057  2023 ;
                     1057  2024 ; COMPLETION CODES:
                     1057  2025 ;       IO status in STATUS if error
                     1057  2026 ;
                     1057  2027 ; SIDE EFFECTS:
                     1057  2028 ;       Program exit if error found
                     1057  2029 ;
                     1057  2030 ;--
                     1057  2031 CHECK_IOSB:
0000'8F   0BE6'CF    B1   1057  2032          CMPW    XE_IOSB,#SS$_NORMAL     ; Is the QIO O.K.?
                01   12   105E  2033          BNEQ    10$                     ; Br if not
                05   1060  2034          RSB                             ; Return
                     1061  2035 10$:
7E   0BE6'CF    3C   1061  2036          MOVZWL  XE_IOSB,-(SP)           ; Push the error status code
0769'CF    6E    D0   1066  2037          MOVL    (SP),STATUS             ; Set return status
          01    DD   106B  2038          PUSHL   #1                      ; Argument count
        050F    31   106D  2039          BRW     ERROR_EXIT              ; Error exit
                     1070  2040
```

```
                     1070  2042          .SBTTL  Check QIO AST Routine
                     1070  2043 ;++
                     1070  2044 ; FUNCTIONAL DESCRIPTION:
                     1070  2045 ;       This routine will be called as a QIO completion AST routine
                     1070  2046 ;       It checks IO status block and the AST parameter
                     1070  2047 ;
                     1070  2048 ; CALLING SEQUENCE:
                     1070  2049 ;       Called via AST at $QIO completion
                     1070  2050 ;
                     1070  2051 ; INPUT PARAMETERS:
                     1070  2052 ;       NONE
                     1070  2053 ;
                     1070  2054 ; IMPLICIT INPUTS:
                     1070  2055 ;       NONE
                     1070  2056 ;
                     1070  2057 ; OUTPUT PARAMETERS:
                     1070  2058 ;       NONE
                     1070  2059 ;
                     1070  2060 ; IMPLICIT OUTPUTS:
                     1070  2061 ;       Error message if error
                     1070  2062 ;
                     1070  2063 ; COMPLETION CODES:
                     1070  2064 ;       IO status in STATUS if error
                     1070  2065 ;
                     1070  2066 ; SIDE EFFECTS:
                     1070  2067 ;       Program exit if error
                     1070  2068 ;
                     1070  2069 ;--
                     1070  2070 CHK_QIO_AST:
                OFFC 1070  2071          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                FFE2   30  1072  2072          BSBW    CHECK_IOSB              ; Go check IO status block
04 AC  00000064 8F     D1  1075  2073          CMPL    #PRM,4(AP)              ; Check AST parameter
                01     12  107D  2074          BNEQ    10$                     ; Branch if not #1 (STARTUP)
                   04  107F  2075          RET
                       1080  2076 10$:
0A77'CF         DF     1080  2077          PUSHAL  ASTPAR_ERRMSG           ; Error message
                FF5F   31  1084  2078          BRW     FAIL_OUT
```

```
                     1087  2080          .SBTTL  Remote Loopback I/O Timeout AST Routine
                     1087  2081 ;++
                     1087  2082 ; FUNCTIONAL DESCRIPTION:
                     1087  2083 ;       This routine services the completion AST if during the remote loopback
                     1087  2084 ;       test we time out while waiting for a message to be returned.  It forces
                     1087  2085 ;       the test to continue by setting the event flag for which we were waiting
                     1087  2086 ;       when we timed out.
                     1087  2087 ;
                     1087  2088 ; CALLING SEQUENCE:
                     1087  2089 ;       Called via AST
                     1087  2090 ;
                     1087  2091 ; INPUT PARAMETERS:
                     1087  2092 ;       04(AP) - Ignored
                     1087  2093 ;
                     1087  2094 ; IMPLICIT INPUTS:
                     1087  2095 ;       NONE
                     1087  2096 ;
                     1087  2097 ; OUTPUT PARAMETERS:
                     1087  2098 ;       NONE
                     1087  2099 ;
                     1087  2100 ; IMPLICIT OUTPUTS:
                     1087  2101 ;       NONE
                     1087  2102 ;
                     1087  2103 ; COMPLETION CODES:
                     1087  2104 ;       Status from $SETEF
                     1087  2105 ;
                     1087  2106 ; SIDE EFFECTS:
                     1087  2107 ;       Another timer AST is enabled.
                     1087  2108 ;       Synchronous test continues because event flag is set.
                     1087  2109 ;
                     1087  2110 ;--
                     1087  2111
                     1087  2112 LOOPBACK_TMO:
                0000 1087  2113          .WORD   ^M<>
                     1089  2114
      21 0049'CF  E9 1089  2115          BLBC    FLAG,5$                 ; Skip this msg if normal running...
                     108E  2116                                          ; ...since there will be another anyway
                     108E  2117          $FAO_S  CTRSTR = LOOPBACK_TMO_MSG,-
                     108E  2118                  OUTLEN = BUFFER_PTR,-
                     108E  2119                  OUTBUF = FAO_BUF,-
                     108E  2120                  P1     = #0 ,-
                     108E  2121                  P2     = ITERATION
          0213  30 10A7  2122          BSBW    DUMP_SIGNAL
  1256'CF   00  FB 10AA  2123          CALLS   #0,DUMP_STAMPS
        50  7E  DE 10AF  2124 5$:     MOVAL   -(SP),R0                ; Create STATE holder for following
                     10B2  2125          $READEF_S EFN = #XMIT_EFN,-      ; Check to see...
                     10B2  2126                    STATE = (R0)           ; ...if we hung while trying to transmit
        8E  05  D3 10BD  2127          BITL    #XMIT_EFN,(SP)+         ; Have we transmitted successfully?
                     10C0  2128                                          ; Note: flag is set except while in...
                     10C0  2129                                          ; ...the $QIOW in REMOTE_LOOPBACK_TEST
            0F  12 10C0  2130          BNEQ    10$                     ; BR if transmit is complete - we're...
                     10C2  2131                                          ; ...waiting for recv, and that will
                     10C2  2132                                          ; ...eventually terminate
09 0049'CF  01  E1 10C2  2133          BBC     #TEST_OVERV,FLAG,10$    ; BR if overall timer hasn't rung yet
      056B'CF  DF 10C8  2134          PUSHAL  TIMEOUT9                ; We seem to be hung in transmit. Abort
  13D1'CF  01  FB 10CC  2135          CALLS   #1,TIME_ERR_OUT
                     10D1  2136 10$:
```

```
                10D1  2137          $SETIMR_S DAYTIM = HALFMIN,-     ; Allow time for messages to complete
                10D1  2138                    ASTADR = LOOPBACK_TMO,-
                10D1  2139                    REQIDT = #LOOPBACR_TMO
                10E7  2140          $SETEF_S EFN = #RECV_EFN
          04    10F0  2141          RET
```

```
                     10F1  2143          .SBTTL  Remote Loopback I/O Completion AST Routine
                     10F1  2144 ;++
                     10F1  2145 ; FUNCTIONAL DESCRIPTION:
                     10F1  2146 ;       This routine services the completion AST from the remote loopback test's
                     10F1  2147 ;       attempt to read.  It has to be a bit more forgiving than most routines,
                     10F1  2148 ;       since we may have $CANCELled the I/O due to a timeout.  The timeout
                     10F1  2149 ;       happens occasionally since not all Ethernet messages are delivered.
                     10F1  2150 ;
                     10F1  2151 ; CALLING SEQUENCE:
                     10F1  2152 ;       Called via AST at $QIO READ
                     10F1  2153 ;
                     10F1  2154 ; INPUT PARAMETERS:
                     10F1  2155 ;       04(AP) - Index into SIZE_TBL1 for the size of the message we attempted
                     10F1  2156 ;                to read.  This index may be used for LOOPBACK_ACK as well, a
                     10F1  2157 ;                parallel table.
                     10F1  2158 ;
                     10F1  2159 ; IMPLICIT INPUTS:
                     10F1  2160 ;       RCV_IOSB - $QIO read completion status
                     10F1  2161 ;       SIZE_TBL1 - Table of message sizes
                     10F1  2162 ;       ITERATION - Count of successful messages sent
                     10F1  2163 ;
                     10F1  2164 ; OUTPUT PARAMETERS:
                     10F1  2165 ;       ITERATION - Incremented if this message was successful
                     10F1  2166 ;
                     10F1  2167 ; IMPLICIT OUTPUTS:
                     10F1  2168 ;       LOOPBACK_ACK - Bit set to indicate that a particular message size was
                     10F1  2169 ;                received
                     10F1  2170 ;
                     10F1  2171 ; COMPLETION CODES:
                     10F1  2172 ;       Status from RECV_AST
                     10F1  2173 ;
                     10F1  2174 ; SIDE EFFECTS:
                     10F1  2175 ;       Message and program exit if error found
                     10F1  2176 ;
                     10F1  2177 ;--
                     10F1  2178
                     10F1  2179 LOOPBACK_AST:
                0OC0 10F1  2180          .WORD   ^M<R6,R7>
                     10F3  2181
      56   04 AC   D0 10F3  2182          MOVL    04(AP),R6               ; Get index into SIZE_TBL1
   57   0BFC'CF    D0 10F7  2183          MOVL    RECV_BUF_ADR,R7         ; Get pointer to message buffer
        0A A7      DD 10FC  2184          PUSHL   10(R7)
   7E   0BF0'CF    3C 10FF  2185          MOVZWL  RCV_IOSB+2,-(SP)
   1221'CF   02    FB 1104  2186          CALLS   #2,RECV_STAMP           ; Remember message received if dump mode
0BEE'CF   0000'8F  B1 1109  2187          CMPW    #SS$_ABORT,RCV_IOSB     ; Did we $CANCEL this transmission?
             50    13 1110  2188          BEQL    20$                     ; BR if so, no point in normal checks
      3B 0049'CF   E9 1112  2189          BLBC    FLAG,10$                ; Skip sequence check if normal running
 0A A7   0791'CF   D1 1117  2190          CMPL    ITERATION,10(R7)        ; Check message sequence number...
             33    13 111D  2191          BEQL    10$                     ; ...and skip if it's OK
                      111F  2192          $FAO_S  CTRSTR = OUT_OF_SEQ_MSG,-
                      111F  2193                  OUTLEN = BUFFER_PTR,-
                      111F  2194                  OUTBUF = FAO_BUF,-
                      111F  2195                  P1     = ITERATION,-
                      111F  2196                  P2     = SIZE_TBL1[R6],-
                      111F  2197                  P3     = 10(R7),-
                      111F  2198                  P4     = RCV_IOSB+2,-
                      111F  2199                  P5     = #0
```

```
                0176   30  1144  2200          BSBW    DUMP_SIGNAL
    1256'CF     00     FB  1147  2201          CALLS   #0,DUMP_STAMPS          ; List the last several msgs sent & recv
0A  A7   0791'CF       D0  114C  2202          MOVL    ITERATION,10(R7)       ; Fix up sequence number...
                           1152  2203                                         ; ...so data comparison can be done
                           1152  2204 10$:
    0C54'C6     01     88  1152  2205          BISB2   #1,LOOPBACK_ACK(R6)     ; Flag that this size was received
      099E'CF46        DD  1157  2206          PUSHL   SIZE_TBL1[R6]          ; Now call general routine...
    1182'CF     01     FB  115C  2207          CALLS   #1,RECV_AST            ; ...to check message's correctness
                       04  1161  2208          RET
                           1162  2209
                           1162  2210 20$:     $FAO_S  CTRSTR = ABORT_MSG,-
                           1162  2211                 OUTLEN = BUFFER_PTR,-
                           1162  2212                 OUTBUF = FAO_BUF,-
                           1162  2213                 P1     = ITERATION
                0141   30  1179  2214          BSBW    DUMP_SIGNAL
    1256'CF     00     FB  117C  2215          CALLS   #0,DUMP_STAMPS          ; List the last several msgs sent & recv
                       04  1181  2216          RET
```

```
                     1182  2218          .SBTTL  Receive data AST routine
                     1182  2219 ;++
                     1182  2220 ; FUNCTIONAL DESCRIPTION:
                     1182  2221 ;       This routine will be called as receive data AST routine
                     1182  2222 ;       It checks IO status and compare the data in the receive buffer
                     1182  2223 ;       against the transmit buffer
                     1182  2224 ;
                     1182  2225 ; CALLING SEQUENCE:
                     1182  2226 ;       Called via AST at $QIO READ
                     1182  2227 ;
                     1182  2228 ; INPUT PARAMETERS:
                     1182  2229 ;       AST parameter = message length
                     1182  2230 ;
                     1182  2231 ; IMPLICIT INPUTS:
                     1182  2232 ;       NONE
                     1182  2233 ;
                     1182  2234 ; OUTPUT PARAMETERS:
                     1182  2235 ;       NONE
                     1182  2236 ;
                     1182  2237 ; IMPLICIT OUTPUTS:
                     1182  2238 ;       Error message if error found
                     1182  2239 ;
                     1182  2240 ; COMPLETION CODES:
                     1182  2241 ;       in STATUS
                     1182  2242 ;
                     1182  2243 ; SIDE EFFECTS:
                     1182  2244 ;       Program exit if error found
                     1182  2245 ;
                     1182  2246 ;--
                     1182  2247 RECV_AST:
                0FFC 1182  2248          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     1184  2249
          54  04 AC  D0   1184  2250          MOVL    4(AP),R4                ; Message length as ast param
     55   0BFC'CF   D0   1188  2251          MOVL    RECV_BUF_ADR,R5         ; Address of receive buffer
     56   0BF8'CF   D0   118D  2252          MOVL    XMIT_BUF_ADR,R6         ; Address of transmit buffer
          08  65   B1   1192  2253          CMPW    (R5),#8                 ; Is this an external transfer with a...
                     1195  2254                                              ; ...skip count modification?
                02   12   1195  2255          BNEQ    10$                     ; BR if not
                65   B4   1197  2256          CLRW    (R5)                    ; Reset the skip count to zero
                     1199  2257 10$:
OBEE'CF    0000'8F   B1   1199  2258          CMPW    #SS$_DATAOVERUN,RCV_IOSB ; Is our buffer too small for message?
                09   13   11A0  2259          BEQL    12$                     ; Length problems are recoverable (below)
OBEE'CF    0000'8F   B1   11A2  2260          CMPW    #SS$_NORMAL,RCV_IOSB    ; Is the QIO O.K.?
                34   12   11A9  2261          BNEQ    20$                     ; BR if not
     54    0BF0'CF   B1   11AB  2262 12$:     CMPW    RCV_IOSB+2,R4           ; Did we get an entire message?
                26   13   11B0  2263          BEQL    15$                     ; BR if we did
                     11B2  2264          $FAO_S  CTRSTR = BADLEN,-       ; Complain if we did not...
                     11B2  2265                  OUTLEN = BUFFER_PTR,-
                     11B2  2266                  OUTBUF = FAO_BUF,-
                     11B2  2267                  P1     = ITERATION,-
                     11B2  2268                  P2     = RCV_IOSB+2,-   ; ...telling actual size...
                     11B2  2269                  P3     = R4             ; ...versus requested size
                00EB 30   11CF  2270          BSBW    DUMP_SIGNAL
     1256'CF    00   FB   11D2  2271          CALLS   #0,DUMP_STAMPS          ; If in DUMP mode, print a history
                04   11D7  2272          RET                             ; Treat this as a recoverable error
                     11D8  2273 15$:
     66   65   54   29   11D8  2274          CMPC3   R4,(R5),(R6)            ; Compare the data
```

```
                     10   12  11DC  2275          BNEQ    30$
                          04  11DE  2276          RET
                              11DF  2277 20$:
                7E  0BEE'CF   3C  11DF  2278          MOVZWL  RCV_IOSB,-(SP)          ; Push the error status code
               0769'CF   6E   D0  11E4  2279          MOVL    (SP),STATUS             ; Set return status
                         01   DD  11E9  2280          PUSHL   #1                      ; Argument count for the error message
                       0391   31  11EB  2281          BRW     ERROR_EXIT              ; Error exit
                              11EE  2282
                              11EE  2283 30$:
                7E    61   9A  11EE  2284          MOVZBL  (R1),-(SP)              ; Get good byte
                7E    63   9A  11F1  2285          MOVZBL  (R3),-(SP)              ; Get bad byte
                7E    54   50  C3  11F4  2286          SUBL3   R0,R4,-(SP)             ; Get the byte number
    5B  07B8'CF  00000788'8F  C1  11F8  2287          ADDL3   #UNIT_LIST,UNIT_LIST,R11 ; Get the unit block address
                      14  AB  DF  1202  2288          PUSHAL  UETUNT$T_FILSPC(R11)    ; Use it to point to the device name
                          04  DD  1205  2289          PUSHL   #4                      ; Push arg count
                0074801A  8F  DD  1207  2290          PUSHL   #UETP$_DATADEVERR!STS$K_ERROR ; Push the signal name
                          06  DD  120D  2291          PUSHL   #6                      ; Push arg count
                0074801A  8F  D0  120F  2292          MOVL    #UETP$_DATADEVERR!STS$K_ERROR,-
                     0769'CF        1215  2293                  STATUS                  ; Push the signal name
                       0364   31  1218  2294          BRW     ERROR_EXIT              ; Failure exit
```

```
                        121B  2296          .SBTTL  DUMP Mode Routines
                        121B  2297 ;+
                        121B  2298 ; Being a set of routines to keep track of the last several messages sent
                        121B  2299 ; and received during the remote loopback test if running in DUMP mode.
                        121B  2300 ; To call XMIT_STAMP ($QIO transmit time stamp) or RECV_STAMP (AST received
                        121B  2301 ; time stamp):
                        121B  2302 ;           PUSHL   MESSAGE-SEQUENCE-NUMBER
                        121B  2303 ;           PUSHL   BUFFER-LENGTH
                        121B  2304 ;           CALLS   #1,{XMIT,RECV}_STAMP
                        121B  2305 ;
                        121B  2306 ; To call DUMP_STAMPS (dump recent time stamps):
                        121B  2307 ;           CALLS   #0,DUMP_STAMPS
                        121B  2308
                        121B  2309          .ENABLE LSB
                        121B  2310 XMIT_STAMP:
                    0004121B  2311          .WORD   ^M<R2>
                  50    D4  121D  2312          CLRL    R0                      ; Remember that we're XMIT_STAMP
                  05    11  121F  2313          BRB     10$
                        1221  2314 RECV_STAMP:
                    0004  1221  2315          .WORD   ^M<R2>
              50    01    D0  1223  2316          MOVL    #1,R0                   ; Remember that we're RECV_STAMP
          52  0C58'CF    D0  1226  2317 10$:     MOVL    SBP,R2                  ; Get pointer into time stamp list
              82    50    B0  122B  2318          MOVW    R0,(R2)+                ; Save RECV/XMIT...
            82    04 AC   B0  122E  2319          MOVW    04(AP),(R2)+            ; ...length of message...
            82    08 AC   D0  1232  2320          MOVL    08(AP),(R2)+            ; ...message sequence number...
                        1236  2321          $GETTIM_S TIMADR = (R2)              ; ...and time when sent or received
              52    08    C0  123F  2322          ADDL2   #8,R2                   ; Point to the next time stamp slot
      52  00000D5C'8F    D1  1242  2323          CMPL    #STAMP_BUFFER+TSCLL,R2  ; It's a circular list - have we...
                  05    14  1249  2324          BGTR    20$                     ; ...gone around?  BR if we have not
          52  0C5C'CF    DE  124B  2325          MOVAL   STAMP_BUFFER,R2         ; Reuse slots at beginning if we have
        0C58'CF   52    D0  1250  2326 20$:     MOVL    R2,SBP                  ; Save pointer into time stamp list
                  04    1255  2327          RET
                        1256  2328          .DISABLE LSB
                        1256  2329
                        1256  2330 DUMP_STAMPS:
                    0004  1256  2331          .WORD   ^M<R2>
            5F 0049'CF    E9  1258  2332          BLBC    FLAG,30$                ; Skip dump if not in DUMP mode
                        125D  2333          $FAO_S  CTRSTR = TIME_STAMP_INTRO_MSG,-
                        125D  2334                  OUTLEN = BUFFER_PTR,-
                        125D  2335                  OUTBUF = FAO_BUF,-
                        125D  2336                  P1     = RCV_IOSB,-
                        125D  2337                  P2     = RCV_IOSB+4
                  43    10  1278  2338          BSBB    DUMP_SIGNAL             ; Indicate what's to follow
          52  0C58'CF    D0  127A  2339          MOVL    SBP,R2                  ; Pick up pointer into time stamp list
              52    10    C2  127F  2340 10$:     SUBL2   #16,R2                  ; Point to previous slot
      52  00000C4C'8F    D1  1282  2341          CMPL    #STAMP_BUFFER-16,R2     ; Are we at the start of the list?
                  05    19  1289  2342          BLSS    20$                     ; BR if not - slot is legit
          52  0D4C'CF    DE  128B  2343          MOVAL   STAMP_BUFFER+TSCLL-16,R2 ; Set pointer to last slot
          0C58'CF   52    D1  1290  2344 20$:     CMPL    R2,SBP                  ; Have we gone full circle yet?
                  25    13  1295  2345          BEQL    30$                     ; BR if we have - exit routine
            50    08 A2   DE  1297  2346          MOVAL   8(R2),R0                ; Get address of time stamp in slot
                        129B  2347          $FAO_S  CTRSTR = TIME_STAMP_MSG,-
                        129B  2348                  OUTLEN = BUFFER_PTR,-
                        129B  2349                  OUTBUF = FAO_BUF,-
                        129B  2350                  P1     = (R2),-
                        129B  2351                  P2     = 2(R2),-
                        129B  2352                  P3     = 4(R2),-
```

```
                     129B  2353                    P4      = R0
                03   10   12B8  2354            BSBB    DUMP_SIGNAL             ; Type one time stamp...
                C3   11   12BA  2355            BRB     10$                     ; ...and loop for the next
                     04   12BC  2356 30$:       RET
                     12BD  2357
                     12BD  2358 DUMP_SIGNAL:
          0053'CF    DF   12BD  2359            PUSHAL  BUFFER_PTR
                01   DD   12C1  2360            PUSHL   #1
          00741133   8F   DD   12C3  2361            PUSHL   #UETP$_TEXT!STS$K_INFO
    00000000'GF      03   FB   12C9  2362            CALLS   #3,G^LIB$SIGNAL
                     05   12D0  2363            RSB
```

```
                                        12D1  2365          .SBTTL  COUNTER_CHECK Routine
                                        12D1  2366 ;++
                                        12D1  2367 ; FUNCTIONAL DESCRIPTION:
                                        12D1  2368 ;       This routine is the UNA counters checking routine. A list of
                                        12D1  2369 ;       specific UNA counters which are considered to indicate errors
                                        12D1  2370 ;       is referenced at CNTR_TBL. When the counters are read their
                                        12D1  2371 ;       value is checked to see if it is non zero.  If non zero a warning
                                        12D1  2372 ;       message will be printed containing the name of the counter and
                                        12D1  2373 ;       it's value.
                                        12D1  2374 ;
                                        12D1  2375 ; CALLING SEQUENCE:
                                        12D1  2376 ;       PUSHAL  COUNTER_DESC
                                        12D1  2377 ;       CALLS   #1, COUNTER_CHECK
                                        12D1  2378 ;
                                        12D1  2379 ; INPUT PARAMETERS:
                                        12D1  2380 ;       4(AP) = Address of the counters to check.
                                        12D1  2381 ;
                                        12D1  2382 ; IMPLICIT INPUTS:
                                        12D1  2383 ;       NONE
                                        12D1  2384 ;
                                        12D1  2385 ; OUTPUT PARAMETERS:
                                        12D1  2386 ;       Sets the error occured flag if a warning message is printed.
                                        12D1  2387 ;
                                        12D1  2388 ; IMPLICIT OUTPUTS:
                                        12D1  2389 ;       Error or success messages
                                        12D1  2390 ;
                                        12D1  2391 ; COMPLETION CODES:
                                        12D1  2392 ;       NONE
                                        12D1  2393 ;
                                        12D1  2394 ; SIDE EFFECTS:
                                        12D1  2395 ;       NONE
                                        12D1  2396 ;
                                        12D1  2397 ;--
                                        12D1  2398
                                        12D1  2399 COUNTER_CHECK:
                                  0FFC  12D1  2400          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                        12D3  2401
                              56  04 AC   D0  12D3  2402          MOVL    4(AP),R6                ; Get the descriptor address
                                 57  66   3C  12D7  2403          MOVZWL  (R6),R7                 ; Get the buffer size
                  0BE2'CF     04 A6  57   C1  12DA  2404          ADDL3   R7,4(R6),END_ADR        ; Save the buffer end address
                              56  04 A6   D0  12E1  2405          MOVL    4(R6),R6                ; Get the buffer address
                           59  088D'CF     DE  12E5  2406          MOVAL   CNTR_TBL,R9             ; Set table address
                                        12EA  2407 CHK_LOOP:
                           0BE2'CF     56   D1  12EA  2408          CMPL    R6,END_ADR              ; All done?
                                       03   12  12EF  2409          BNEQ    10$                     ; BR if not...
                                    00DC    31  12F1  2410          BRW     COUNTER_EXIT            ; ...else bail out
                                        12F4  2411 10$:
                                 58  86   3C  12F4  2412          MOVZWL  (R6)+,R8                ; Get cntrl desc
                                     00   EF  12F7  2413          EXTZV   #NMA$V_CNT_TYP,-
                                     0C       12F9  2414                  #NMA$S_CNT_TYP,-
                                 58  58       12FA  2415                  R8,R8                   ; Get the counter type
                                        12FC  2416 20$:
                                 5A  89   3C  12FC  2417          MOVZWL  (R9)+,R10               ; Get a table code
                                     00   EF  12FF  2418          EXTZV   #NMA$V_CNT_TYP,-
                                     0C       1301  2419                  #NMA$S_CNT_TYP,-
                                 5B  5A       1302  2420                  R10,R11                 ; Get the counter type
                                 5B  58   D1  1304  2421          CMPL    R8,R11                  ; Is this it?
```

```
                4B   13  1307  2422          BEQL    80$                     ; BR if yes
            59  04   C0  1309  2423          ADDL2   #4,R9                   ; Skip name pointer
000008B7'   8F   59   D1  130C  2424          CMPL    R9,#TBL_END             ; End of table?
                E7   12  1313  2425          BNEQ    20$                     ; BR if not
     59  088D'CF   DE  1315  2426          MOVAL   CNTR_TBL,R9             ; Set table address
                0D   EF  131A  2427          EXTZV   #NMA$V_CNT_WID,-
                02       131C  2428                  #NMA$S_CNT_WID,-
     58    FE A6       131D  2429                  -2(R6),R8               ; Get the counter width
03 FE A6    0C   E1  1320  2430          BBC     #NMA$V_CNT_MAP,-2(R6),30$ ; If not a mapped counter then carry on el
            56   02  1325  2431          ADDL2   #2,R6                   ; ...skip the map word
                     1328  2432 30$:
02   01   58   8F  1328  2433          CASEB   R8,#1,#2                ; Skip the counter
                     132C  2434 40$:
                0017' 132C  2435          .WORD   50$-40$
                001C' 132E  2436          .WORD   60$-40$
                0022' 1330  2437          .WORD   70$-40$
        029A'CF   DF  1332  2438          PUSHAL  CASE_FAILED             ; Push the string address
              01  DD  1336  2439          PUSHL   #1                      ; Push the argument count
 00741132     8F  DD  1338  2440          PUSHL   #UETP$_TEXT!STS$K_ERROR ; Push the signal name
              03  DD  133E  2441          PUSHL   #3                      ; Push the argument count
          023C    31  1340  2442          BRW     ERROR_EXIT              ; Thats it
                     1343  2443 50$:
              56  D6  1343  2444          INCL    R6                      ; Skip a byte counter
          FFA2    31  1345  2445          BRW     CHK_LOOP
                     1348  2446 60$:
           56  02  C0  1348  2447          ADDL2   #2,R6                   ; Skip a word counter
          FF9C    31  134B  2448          BRW     CHK_LOOP
                     134E  2449 70$:
           56  04  C0  134E  2450          ADDL2   #4,R6                   ; Skip a long word counter
          FF96    31  1351  2451          BRW     CHK_LOOP
                     1354  2452 80$:
              0D   EF  1354  2453          EXTZV   #NMA$V_CNT_WID,-
                02       1356  2454                  #NMA$S_CNT_WID,-
     58    FE A6       1357  2455                  -2(R6),R8               ; Get the counter width
03 FE A6    0C   E1  135A  2456          BBC     #NMA$V_CNT_MAP,-2(R6),90$ ; If not a mapped counter then carry on el
            56    2  C0  135F  2457          ADDL2   #2,R6                   ; ...skip the map word
                     1362  2458 90$:
02   01   58   8F  1362  2459          CASEB   R8,#1,#2                ; Skip the counter
                     1366  2460 100$:
                0017' 1366  2461          .WORD   110$-100$
                001F' 1368  2462          .WORD   120$-100$
                0027' 136A  2463          .WORD   130$-100$
        029A'CF   DF  136C  2464          PUSHAL  CASE_FAILED             ; Push the string address
              01  DD  1370  2465          PUSHL   #1                      ; Push the argument count
 00741132     8F  DD  1372  2466          PUSHL   #UETP$_TEXT!STS$K_ERROR ; Push the signal name
              03  DD  1378  2467          PUSHL   #3                      ; Push the argument count
          0202    31  137A  2468          BRW     ERROR_EXIT              ; Thats it
                     137D  2469 110$:
OBDE'CF       86   9A  137D  2470          MOVZBL  (R6)+,COUNTER           ; Get a byte counter
          000D    31  1382  2471          BRW     140$
                     1385  2472 120$:
OBDE'CF       86   3C  1385  2473          MOVZWL  (R6)+,COUNTER           ; Get a word counter
          0005    31  138A  2474          BRW     140$
                     138D  2475 130$:
OBDE'CF       86   D0  138D  2476          MOVL    (R6)+,COUNTER           ; Get a long word counter
                     1392  2477 140$:
              08   12  1392  2478          BNEQ    150$
```

```
                59   088D'CF    DE  1394  2479          MOVAL   CNTR_TBL,R9             ; Reset the table address
                       FF4E     31  1399  2480          BRW     CHK_LOOP                ; BR if counter was zero
                                    139C  2481 150$:
                                    139C  2482          $FAO_S  CTRSTR = COUNTER_MSG,-  ; Generate a bad counter message
                                    139C  2483                  OUTLEN = BUFFER_PTR,-
                                    139C  2484                  OUTBUF = FAO_BUF,-
                                    139C  2485                  P1     = (R9)+,-
                                    139C  2486                  P2     = COUNTER
                         0053'CF    DF  13B5  2487          PUSHAL  BUFFER_PTR              ; Push the string address
                              01    DD  13B9  2488          PUSHL   #1                      ; Push the argument count
                        00741133    8F  DD  13BB  2489      PUSHL   #UETP$_TEXT!STS$K_INFO  ; Push the signal name
                00000000'GF    03   FB  13C1  2490          CALLS   #3, G^LIB$SIGNAL        ; Print the error message
                59   088D'CF    DE  13C8  2491          MOVAL   CNTR_TBL,R9             ; Reset the counter table pointer
                       FF1A     31  13CD  2492          BRW     CHK_LOOP                ; Thats it
                                    13D0  2493
                                    13D0  2494 COUNTER_EXIT:
                                04  13D0  2495          RET
```

```
                     13D1  2497          .SBTTL  Half Minute Timer Expiration Routine
                     13D1  2498 ;++
                     13D1  2499 ; FUNCTIONAL DESCRIPTION:
                     13D1  2500 ;       This routine will be called only if the timer which was set to prevent
                     13D1  2501 ;       program hangs goes off.
                     13D1  2502 ;
                     13D1  2503 ; CALLING SEQUENCE:
                     13D1  2504 ;       Called via AST at $SETIMR expiration.
                     13D1  2505 ;
                     13D1  2506 ; INPUT PARAMETERS:
                     13D1  2507 ;       NONE
                     13D1  2508 ;
                     13D1  2509 ; IMPLICIT INPUTS:
                     13D1  2510 ;       NONE
                     13D1  2511 ;
                     13D1  2512 ; OUTPUT PARAMETERS:
                     13D1  2513 ;       NONE
                     13D1  2514 ;
                     13D1  2515 ; IMPLICIT OUTPUTS:
                     13D1  2516 ;       Time out error message
                     13D1  2517 ;
                     13D1  2518 ; COMPLETION CODES:
                     13D1  2519 ;       #SS$_TIMEOUT
                     13D1  2520 ;
                     13D1  2521 ; SIDE EFFECTS:
                     13D1  2522 ;       Program exit
                     13D1  2523 ;
                     13D1  2524 ;--
                     13D1  2525
                     13D1  2526 TIME_ERR_OUT:
             0FFC    13D1  2527          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
 00000000'8F   DD    13D3  2528          PUSHL   #SS$_TIMEOUT                ; Push the signal name = timeout
0769'CF    6E   D0  13D9  2529          MOVL    (SP),STATUS                 ; Set exit status
       04 AC    DD  13DE  2530          PUSHL   4(AP)                       ; Push the REQIDT which is the timeout
                     13E1  2531                                              ; string address
       01       DD  13E1  2532          PUSHL   #1                          ; Push the argument count total
 00741132 8F    DD  13E3  2533          PUSHL   #UETP$_TEXT!STS$K_ERROR     ; Push the message number
       04       DD  13E9  2534          PUSHL   #4                          ; Push the argument count
      0191      31  13EB  2535          BRW     ERROR_EXIT                  ; Bail out completely
```

```
                     13EE  2537          .SBTTL  Test End Timer Expiration Routine
                     13EE  2538 ;++
                     13EE  2539 ; FUNCTIONAL DESCRIPTION:
                     13EE  2540 ;       This routine will be called only if the timer which was set to
                     13EE  2541 ;       terminate the test goes off.
                     13EE  2542 ;
                     13EE  2543 ; CALLING SEQUENCE:
                     13EE  2544 ;       Called via AST at $SETIMR expiration.
                     13EE  2545 ;
                     13EE  2546 ; INPUT PARAMETERS:
                     13EE  2547 ;       NONE
                     13EE  2548 ;
                     13EE  2549 ; IMPLICIT INPUTS:
                     13EE  2550 ;       NONE
                     13EE  2551 ;
                     13EE  2552 ; OUTPUT PARAMETERS:
                     13EE  2553 ;       NONE
                     13EE  2554 ;
                     13EE  2555 ; IMPLICIT OUTPUTS:
                     13EE  2556 ;       NONE
                     13EE  2557 ;
                     13EE  2558 ; COMPLETION CODES:
                     13EE  2559 ;       NONE
                     13EE  2560 ;
                     13EE  2561 ; SIDE EFFECTS:
                     13EE  2562 ;       NONE
                     13EE  2563 ;
                     13EE  2564 ;--
                     13EE  2565
                     13EE  2566 TIME_SUC_OUT:
                OFFC 13EE  2567          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     13F0  2568
0049'CF    02   A8  13F0  2569          BISW2   #TEST_OVERM,FLAG                    ; Set test over flag
                04   13F5  2570          RET
                     13F6  2571
```

```
                        .SBTTL  System Service Exception Handler
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;       This routine is executed if a software or hardware exception occurs or
                ;       if a LIB$SIGNAL system service is used to output a message.
                ;       Information about this method of handling messages and errors can be
                ;       found in the VMS COMMON RUN-TIME manual and in the VMS SYSTEM SERVICE
                ;       manual.
                ;
                ; CALLING SEQUENCE:
                ;       Entered via an exception from the system
                ;
                ; INPUT PARAMETERS:
                ;       ERROR_COUNT    = previous cumulative error count
                ;
                ;          AP ---->     +---------------+
                ;                       |       2       |
                ;                       |---------------|
                ;                       | SIGNL ARY PNT |
                ;                       |---------------|
                ;                       | MECH  ARY PNT |
                ;                       |---------------|  ---------
                ;                       |       4       |         ^
                ;                       |---------------|         |
                ;                       | ESTABLISH FP  |         |
                ;                       |---------------|         |
                ;                       |    DEPTH      | Mechanism Array
                ;                       |---------------|         |
                ;                       |      R0       |         |
                ;                       |---------------|         |
                ;                       |      R1       |         v
                ;                       |---------------|  ---------
                ;                       |       N       |         ^
                ;                       |---------------|         |
                ;                       | CONDITION NAME|         |
                ;                       |---------------|         |
                ;                       | N-3 ADDITIONAL|    Signal Array
                ;                       | LONG WORD ARGS|         |
                ;                       |---------------|         |
                ;                       |      PC       |         |
                ;                       |---------------|         |
                ;                       |      PSL      |         v
                ;                       |---------------|  ---------
                ; IMPLICIT INPUTS:
                ;       NONE
                ;
                ; OUTPUT PARAMETERS:
                ;       NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;       NONE
                ;
                ; COMPLETION CODES:
                ;       NONE
                ;
                ; SIDE EFFECTS:
                ;       May branch to ERROR_EXIT
                ;--
```

```
                                13F6  2630
                                13F6  2631 SSERROR:
                           0FFC 13F6  2632         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                13F8  2633
                                13F8  2634         $SETAST_S ENBFLG = #0                ; Disable AST delivery
                          01 DD 1401  2635         PUSHL   #1                           ; Assume ASTs were enabled
                       50 00' D1 1403  2636         CMPL    S^#SS$_WASSET,R0             ; Were ASTs enabled?
                          02 13 1406  2637         BEQL    10$                          ; BR if they were
                          6E D4 1408  2638         CLRL    (SP)                         ; Set ASTs to remain disabled
                                140A  2639 10$:
                                140A  2640         $SETSFM_S ENBFLG = #0                ; Disable SS failure mode
                          01 DD 1413  2641         PUSHL   #1                           ; Assume SS failure mode was enabled
                       50 00' D1 1415  2642         CMPL    S^#SS$_WASSET,R0             ; Was SS failure mode enabled?
                          02 13 1418  2643         BEQL    20$                          ; BR if it was
                          6E D4 141A  2644         CLRL    (SP)                         ; Set SS failure mode to remain off
                                141C  2645 20$:
                    56    04 AC D0 141C  2646         MOVL    CHF$L_SIGARGLST(AP),R6       ; Get the signal array pointer
                    59    04 A6 7D 1420  2647         MOVQ    CHF$L_SIG_NAME(R6),R9        ; Get NAME in R9 and ARG1 in R10
                             10 ED 1424  2648         CMPZV   #STS$V_FAC_NO,-              ; Is this a message from LIB$SIGNAL?
                                1426  2649                 #STS$S_FAC_NO,-
              00000074 8F    59       1427  2650                 R9,#UETP$_FACILITY
                             14 12 142D  2651         BNEQ    30$                          ; BR if this is not a UETP exception
                       66    02 C2 142F  2652         SUBL2   #2,CHF$L_SIG_ARGS(R6)        ; Drop the PC and PSL
                                1432  2653         $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
                             21 11 1441  2654         BRB     40$                          ; Restore ASTs and SS fail mode
                                1443  2655 30$:
          59  00000000'8F    D1 1443  2656         CMPL    #SS$_SSFAIL,R9               ; RMS failures are SysSvc failures
                             32 12 144A  2657         BNEQ    50$                          ; BR if this can't be an RMS failure
                             10 ED 144C  2658         CMPZV   #STS$V_FAC_NO,-              ; Is it an RMS failure?
                                144E  2659                 #STS$S_FAC_NO,-
                       01    5A       144F  2660                 R10,#RMS$_FACILITY
                             2B 12 1451  2661         BNEQ    50$                          ; BR if not
          5A    F0000000 8F    CA 1453  2662         BICL2   #^XF0000000,R10              ; Strip control bits from status code
                 08 A6    04 39 145A  2663         MATCHC  #4,CHF$L_SIG_ARG1(R6),-      ; Is it an RMS failure for which...
                             14       145E  2664                 #NRAT_LENGTH,-
                    005E'CF          145F  2665                 NO_RMS_AST_TABLE         ; ...no AST can be delivered?
                             1A 13 1462  2666         BEQL    50$                          ; BR if so - must give error here
                                1464  2667 40$:
                             01 BA 1464  2668         POPR    #^M<R0>                      ; Restore SS failure mode...
                                1466  2669         $SETSFM_S ENBFLG = R0                ; ...
                             01 BA 146F  2670         POPR    #^M<R0>                      ; Restore AST enable...
                                1471  2671         $SETAST_S ENBFLG = R0                ; ...
                       50    00' D0 147A  2672         MOVL    S^#SS$_NORMAL,R0             ; Supply a standard status for exit
                             04 147D  2673         RET                                  ; Resume processing (or goto RMS_ERROR)
                                147E  2674 50$:
                 0769'CF     59 D0 147E  2675         MOVL    R9,STATUS                    ; Save the status
                             58 D4 1483  2676         CLRL    R8                           ; Assume for now it's not SS failure
          59  00000000'8F    D1 1485  2677         CMPL    #SS$_SSFAIL,R9               ; But is it a System Service failure?
                             38 12 148C  2678         BNEQ    70$                          ; BR if not - no special case message
                                148E  2679         $GETMSG_S MSGID = R10,-              ; Get SS failure code associated text
                                148E  2680                  MSGLEN = BUFFER_PTR,-
                                148E  2681                  BUFADR = FAO_BUF,-
                                148E  2682                  FLAGS  = #14,-
                                148E  2683                  OUTADR = MSG_BLOCK
                    079A'CF     95 14A5  2684         TSTB    MSG_BLOCK+1                  ; Get FAO arg count for SS failure code
                             16 13 14A9  2685         BEQL    60$                          ; Don't use $GETMSG if no $FAO args...
                    0053'CF     DF 14AB  2686         PUSHAL  BUFFER_PTR                   ; ...else build up...
```

```
                    01   DD  14AF  2687          PUSHL   #1                              ; ...a message describing...
            00741130 8F   DD  14B1  2688          PUSHL   #UETP$_TEXT                     ; ...why the System Service failed
                00   5A   F0  14B7  2689          INSV    R10,#STS$V_SEVERITY,-           ; Give the message...
                6E   03        14BA  2690                  #STS$S_SEVERITY,(SP)            ; ...the correct severity code
                58   03   D0  14BC  2691          MOVL    #3,R8                           ; Count the number of args we pushed
                     05   11  14BF  2692          BRB     70$
                                14C1  2693 60$:
                     5A   DD  14C1  2694          PUSHL   R10                             ; Save SS failure code
                58   01   D0  14C3  2695          MOVL    #1,R8                           ; Count the number of args we pushed
                                14C6  2696 70$:
            57  66   04   C5  14C6  2697          MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
                5E   57   C2  14CA  2698          SUBL2   R7,SP                           ; Save the current signal array...
        6E   04 A6   57   28  14CD  2699          MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
            7E  66   58   C1  14D2  2700          ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
                    00A6  31  14D6  2701          BRW     ERROR_EXIT
```

```
                        14D9  2703          .SBTTL  RMS Error Handler
                        14D9  2704 ;++
                        14D9  2705 ; FUNCTIONAL DESCRIPTION:
                        14D9  2706 ;       This routine handles error returns from RMS calls.
                        14D9  2707 ;
                        14D9  2708 ; CALLING SEQUENCE:
                        14D9  2709 ;       Called by RMS when a file processing error is found.
                        14D9  2710 ;
                        14D9  2711 ; INPUT PARAMETERS:
                        14D9  2712 ;       NONE
                        14D9  2713 ;
                        14D9  2714 ; IMPLICIT INPUTS:
                        14D9  2715 ;       The FAB or RAB associated with the RMS call.
                        14D9  2716 ;
                        14D9  2717 ; OUTPUT PARAMETERS:
                        14D9  2718 ;       NONE
                        14D9  2719 ;
                        14D9  2720 ; IMPLICIT OUTPUTS:
                        14D9  2721 ;       Error message
                        14D9  2722 ;
                        14D9  2723 ; COMPLETION CODES:
                        14D9  2724 ;       NONE
                        14D9  2725 ;
                        14D9  2726 ; SIDE EFFECTS:
                        14D9  2727 ;       Program may exit, depending on severity of the error.
                        14D9  2728 ;
                        14D9  2729 ;--
                        14D9  2730
                        14D9  2731 RMS_ERROR:
                   0FFC 14D9  2732          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        14DB  2733
          56   04 AC   D0  14DB  2734          MOVL    4(AP),R6                    ; See whether we're dealing with...
             66   03   91  14DF  2735          CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
                  16   12  14E2  2736          BNEQ    10$                         ; BR if it's a RAB
       57 09E6'CF   DE  14E4  2737          MOVAL   FILE,R7                     ; FAB-specific code:  text string...
          58   56   D0  14E9  2738          MOVL    R6,R8                       ; ...address of FAB...
            0C A6   DD  14EC  2739          PUSHL   FAB$L_STV(R6)               ; ...STV field for error...
            08 A6   DD  14EF  2740          PUSHL   FAB$L_STS(R6)               ; ...STS field for error...
0769'CF     08 A6   D0  14F2  2741          MOVL    FAB$L_STS(R6),STATUS        ; ...and save the error code
                  15   11  14F8  2742          BRB     COMMON                      ; FAB and RAB share other code
                        14FA  2743 10$:
       57 09F2'CF   DE  14FA  2744          MOVAL   RECORD,R7                   ; RAB-specific code:  text string...
          58   3C A6   D0  14FF  2745          MOVL    RAB$L_FAB(R6),R8            ; ...address of associated FAB...
            0C A6   DD  1503  2746          PUSHL   RAB$L_STV(R6)               ; ...STV field for error...
            08 A6   DD  1506  2747          PUSHL   RAB$L_STS(R6)               ; ...STS field for error...
0769'CF     08 A6   D0  1509  2748          MOVL    RAB$L_STS(R6),STATUS        ; ...and save the error code
                        150F  2749 COMMON:
          5A   34 A8   9A  150F  2750          MOVZBL  FAB$B_FNS(R8),R10           ; Get the file name size
                        1513  2751          $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                        1513  2752                  OUTLEN = BUFFER_PTR,-
                        1513  2753                  OUTBUF = FAO_BUF,-
                        1513  2754                  P1     = R7,-
                        1513  2755                  P2     = R10,-
                        1513  2756                  P3     = FAB$L_FNA(R8)
             0053'CF    DF  152D  2757          PUSHAL  BUFFER_PTR                  ; ...and arguments for ERROR_EXIT...
                   01   DD  1531  2758          PUSHL   #1                          ; ...
          00741130 8F   DD  1533  2759          PUSHL   #UETP$_TEXT                 ; ...
```

```
                00   EF  1539  2760          EXTZV   #STS$V_SEVERITY,-
                03       153B  2761                  #STS$S_SEVERITY,-
          59  0769'CF       153C  2762                  STATUS,R9               ; ...get the severity code...
              6E   59   88  1540  2763          BISB2   R9,(SP)                 ; ...and add it into the signal name
                   05   DD  1543  2764          PUSHL   #5                      ; Current arg count
                0037    31  1545  2765          BRW     ERROR_EXIT
```

```
                     1548  2767          .SBTTL  CTRL/C Handler
                     1548  2768 ;++
                     1548  2769 ; FUNCTIONAL DESCRIPTION:
                     1548  2770 ;       This routine handles CTRL/C AST's
                     1548  2771 ;
                     1548  2772 ; CALLING SEQUENCE:
                     1548  2773 ;       Called via AST
                     1548  2774 ;
                     1548  2775 ; INPUT PARAMETERS:
                     1548  2776 ;       NONE
                     1548  2777 ;
                     1548  2778 ; IMPLICIT INPUTS:
                     1548  2779 ;       NONE
                     1548  2780 ;
                     1548  2781 ; OUTPUT PARAMETERS:
                     1548  2782 ;       NONE
                     1548  2783 ;
                     1548  2784 ; IMPLICIT OUTPUTS:
                     1548  2785 ;       NONE
                     1548  2786 ;
                     1548  2787 ; COMPLETION CODES:
                     1548  2788 ;       NONE
                     1548  2789 ;
                     1548  2790 ; SIDE EFFECTS:
                     1548  2791 ;       NONE
                     1548  2792 ;
                     1548  2793 ;--
                     1548  2794
                     1548  2795 CCASTHAND:
             0FFC    1548  2796          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     154A  2797
 02CF'CF    DF      154A  2798          PUSHAL  CNTRLCMSG                   ; Set message pointer
       01    DD      154E  2799          PUSHL   #1                          ; Set arg count
 00741130    8F      DD    1550  2800    PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
       00    DD      1556  2801          PUSHL   #0                          ; Indicate an abnormal termination
 06C3'CF    DF      1558  2802          PUSHAL  PROCESS_NAME                ; ...
       02    DD      155C  2803          PUSHL   #2                          ; ...
 007410E0    8F      DD    155E  2804    PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
00000000'GF    07    FB    1564  2805    CALLS   #7,G^LIB$SIGNAL             ; Output the message
                     DO    156B  2806    MOVL    #<STS$M_INHIB_MSG!-         ; Set the exit status
                           156C  2807            SS$_CONTROLC-=
                           156C  2808            STS$K_SUCCESS+STS$K_WARNING>,-
0769'CF    0FFFFFFF'8F     156C  2809            STATUS
                           1574  2810    $EXIT_S STATUS                      ; Terminate program cleanly
```

```
                                                .SBTTL  Error Exit
                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;       This routine prints an error message and exits.
                                ;
                                ; CALLING SEQUENCE:
                                ;       MOVx  error status value,STATUS
                                ;       PUSHx error specific information on the stack
                                ;       PUSHL current argument count
                                ;       BRW   ERROR_EXIT
                                ;
                                ; INPUT PARAMETERS:
                                ;       Arguments to LIB$SIGNAL, as above
                                ;
                                ; IMPLICIT INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUT PARAMETERS:
                                ;       Message to SYS$OUTPUT and SYS$ERROR
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       Program exit
                                ;
                                ; COMPLETION CODES:
                                ;       NONE
                                ;
                                ; SIDE EFFECTS:
                                ;       NONE
                                ;
                                ;--

                                ERROR_EXIT:

                                        $SETAST_S ENBFLG = #0                   ; ASTs can play havoc with messages
    15 0049'CF    03    E0      BBS     #BEGIN_MSGV,FLAG,10$                    ; BR if "begin" msg already printed
                  7E    D4      CLRL    -(SP)                                   ; Set the time stamp flag
            000F'CF     DF      PUSHAL  TEST_NAME                               ; Set the test name
                  02    DD      PUSHL   #2                                      ; Push the argument count
        00741039  8F    DD      PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
    00000000'GF   04    FB      CALLS   #4,G^LIB$SIGNAL                         ; Print the startup message
                                10$:
  07AD'CF   08    8E    C1      ADDL3   (SP)+,#8,ARG_COUNT                      ; Get total # args, pop partial count
            0765'CF     D6      INCL    ERROR_COUNT                             ; Keep running error count
                  00    DD      PUSHL   #0                                      ; Push the time parameter
            06C3'CF     DF      PUSHAL  PROCESS_NAME                            ; Push test name...
        000F0002  8F    DD      PUSHL   #^XF0002                                ; ...arg count...
        007410E2  8F    DD      PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
            0765'CF     DD      PUSHL   ERROR_COUNT                             ; Finish off arg list...
            06C3'CF     DF      PUSHAL  PROCESS_NAME                            ; ...
        00010002  8F    DD      PUSHL   #^X10002                                ; ...
        00748022  8F    DD      PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
00000000'GF   07AD'CF   FB      CALLS   ARG_COUNT,G^LIB$SIGNAL                  ; Truly bitch

            0769'CF     D5      TSTL    STATUS                                  ; Did we exit with an error code?
                  09    12      BNEQ    20$                                     ; BR if we did
        007410E2  8F    D0      MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
            0769'CF             STATUS
```

```
                        15EB  2869 20$:
0769'CF   10000000 8F   C8  15EB  2870          BISL    #STS$M_INHIB_MSG,STATUS ; Don't print messages twice!
                        15F4  2871          $EXIT_S STATUS                   ; Exit in error
```

```
                                        .SBTTL  Exit Handler
                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;       This routine handles cleanup at exit.  If the MODE logical name is
                                ;       equated to "ONE", the routine will update the test flag in the
                                ;       UETINIDEV.DAT file depending on the UETUNT$M_TESTABLE flag state in the
                                ;       UETUNT$B_FLAGS field of the unit block for each unit for the device
                                ;       under test.
                                ;
                                ; CALLING SEQUENCE:
                                ;       Invoked automatically by $EXIT System Service.
                                ;
                                ; INPUT PARAMETERS:
                                ;       STATUS  contains the exit status.
                                ;       FLAG    has synchronizing bits.
                                ;       DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
                                ;
                                ; IMPLICIT INPUTS:
                                ;       UNIT_LIST points to the head of a doubly linked circular list of unit
                                ;                 blocks for the device under test.
                                ;
                                ; OUTPUT PARAMETERS:
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       Various files are de-accessed, the process name is reset, and any
                                ;       necessary synchronization with UETPDEV01 is carried out.
                                ;       If the MODE logical name is equated to "ONE", the routine will update
                                ;       the test flag in the UETINIDEV.DAT file depending on the
                                ;       UETUNT$M_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
                                ;       block for each unit for the device under test.
                                ;
                                ; COMPLETION CODES:
                                ;       NONE
                                ;
                                ; SIDE EFFECTS:
                                ;       NONE
                                ;
                                ;--
                                
                                EXIT_HANDLER:
                        0FFC    .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                
                                        $SETSFM_S ENBFLG = #0           ; Turn off System Service failure mode
                                        $SETAST_S ENBFLG = #0           ; We're finished - no more ASTs
                                        $TRNLOG_S LOGNAM = MODE,-       ; Get the run mode
                                                  RSLLEN = BUFFER_PTR,-
                                                  RSLBUF = FAO_BUF
   005B'CF    20    8A   BICB2   #LC_BITM,BUFFER         ; Convert to upper case
005B'CF  4F  8F    91   CMPB    #^A/O/,BUFFER           ; Is this a one shot?
             03    13   BEQL    10$                     ; BR if yes...
           00B8    31   BRW     END_UPDATE              ; ...else don't update UETINIDEV.DAT
                        10$:
03 0049'CF   02    E0   BBS     #SAFE_TO_UPDV,FLAG,20$  ; Only update if it's safe
           00AF    31   BRW     END_UPDATE              ; Else forget it
                        20$:
   5A   0E40'CF    DE   MOVAL   INI_RAB,R10             ; Set the RAB address
```

```
                1E AA   02   90  164A  2930          MOVB    #RAB$C_RFA,RAB$B_RAC(R10) ; Set RFA mode
          10 AA  0E84'CF   06   28  164E  2931          MOVC3   #6,DDB_RFA,RAB$W_RFA(R10) ; Set RFA to DDB line
                                1655  2932          $GET    RAB = (R10)               ; Go back to the DDB record
                   75 50   E9   165E  2933          BLBC    R0,UPDATE_FAILED          ; If failure then forget it
                1E AA   00   90  1661  2934          MOVB    #RAB$C_SEQ,RAB$B_RAC(R10) ; Set back to sequential mode
    5B  07B8'CF  000007B8'8F   C1  1665  2935          ADDL3   #UNIT_LIST,UNIT_LIST,R11 ; Set the unit block list header
                      59   D4  166F  2936          CLRL    R9                        ; Init a counter
                                1671  2937 UNIT_LOOP:
                      01   E1  1671  2938          BBC     #UETUNT$V_TESTABLE,-      ; BR if this unit is not testable
                   02 0B AB         1673  2939                  UETUNT$B_FLAGS(R11),10$
                      59   D6  1676  2940          INCL    R9                        ; Count testable units
                                1678  2941 10$:
                   5B   6B   C0  1678  2942          ADDL2   (R11),R11                 ; Next unit block
             000007B8'8F   5B   D1  167B  2943          CMPL    R11,#UNIT_LIST            ; Are we full circle in the list?
                      ED   12  1682  2944          BNEQ    UNIT_LOOP                 ; BR if not
                      59   D5  1684  2945          TSTL    R9                        ; Any testable units?
                      12   12  1686  2946          BNEQ    20$                       ; BR if yes...
              005F'CF   4E 8F   90  1688  2947          MOVB    #^A/N/,BUFFER+4           ; ...else disable the DDB record...
                                168E  2948          $UPDATE RAB = (R10)               ; ...here
                   3C 50   E9  1697  2949          BLBC    R0,UPDATE_FAILED          ; If error then forget it
                                169A  2950 20$:
                   5B   6B   C0  169A  2951          ADDL2   (R11),R11                 ; Next unit block
             000007B8'8F   5B   D1  169D  2952          CMPL    R11,#UNIT_LIST            ; Are we full circle in the list?
                      4E   13  16A4  2953          BEQL    END_UPDATE                ; BR if yes
                                16A6  2954          $GET    RAB = (R10)               ; Get a record
                   24 50   E9  16AF  2955          BLBC    R0,UPDATE_FAILED          ; If error then forget it
                005B'CF   20   8A  16B2  2956          BICB2   #LC_BITM,BUFFER           ; Convert to uppercase
              005B'CF   55 8F   91  16B7  2957          CMPB    #^A/U/,BUFFER             ; Is it a UCB record?
                      35   12  16BD  2958          BNEQ    END_UPDATE                ; BR if not
                      01   E0  16BF  2959          BBS     #UETUNT$V_TESTABLE,-      ; BR if this unit is testable...
                   D6 0B AB         16C1  2960                  UETUNT$B_FLAGS(R11),20$
              005F'CF   4E 8F   90  16C4  2961          MOVB    #^A/N/,BUFFER+4           ; ...else disable the UCB record...
                                16CA  2962          $UPDATE RAB = (R10)               ; ...here
                   C4 50   E8  16D3  2963          BLBS    R0,20$                    ; Look at the next record if no error
                                16D6  2964 UPDATE_FAILED:
                   0C AA   DD  16D6  2965          PUSHL   RAB$L_STV(R10)            ; Do a simple message...
                      50   DD  16D9  2966          PUSHL   R0                        ; ...to tell of the failure
                00C2'CF   DF  16DB  2967          PUSHAL  INIDEV_UPDERR
                      01   DD  16DF  2968          PUSHL   #1
                      00   EF  16E1  2969          EXTZV   #STS$V_SEVERITY,-         ; Copy the severity from RMS status...
          7E    50   03         16E3  2970                  #STS$S_SEVERITY,R0,-(SP)
    6E  00741130 8F   C8  16E6  2971          BISL2   #UETP$_TEXT,(SP)          ; ...to our message
00000000'GF   05   FB  16ED  2972          CALLS   #5,G^LIB$SIGNAL
                                16F4  2973 END_UPDATE:
                      00   DD  16F4  2974          PUSHL   #0                        ; Set the time flag
                000F'CF   DF  16F6  2975          PUSHAL  TEST_NAME                 ; Push the test name
                      02   DD  16FA  2976          PUSHL   #2                        ; Push arg count
                      00   EF  16FC  2977          EXTZV   #STS$V_SEVERITY,-         ; Push the proper exit severity...
                      03         16FE  2978                  #STS$S_SEVERITY,-
          7E   0769'CF         16FF  2979                  STATUS,-(SP)
    6E  00741080 8F   C8  1703  2980          BISL2   #UETP$_ENDEDD,(SP)        ; ...and use it in our message code
                      04   DD  170A  2981          PUSHL   #4
                   51   5E   D0  170C  2982          MOVL    SP,R1
                                170F  2983          $PUTMSG_S MSGVEC = (R1)           ; Output the message
                                171E  2984          $SETPRN_S PRCNAM = ACNT_NAME      ; Reset the process name
                      04   1729  2985          RET                               ; That's all folks!
                                172A  2986
```

```
                172A  2987          .END    UETUNAS00
```

```
$$.TAB                  = 00000F2C R      03
$$.TABEND               = 00000F70 R      03
$$.TMP                  = 00000000
$$.TMP1                 = 00000001
$$.TMP2                 = 0000006A
$$.TMPX                 = 00000016 R      04
$$.TMPX1                = 0000000D
$$T1                    = 00000000
$$T2                    = 00000006
ABORT_MSG                 000007A2 R      02
ABRT_BRDCST               00000D26 R      05
ACNT_NAME                 00000000 R      02
ADR                       0000087E R      03
ADR1                      000008DC R      03
ADR2                      000008EA R      03
ALL_SET                   000003F7 R      05
ARG_COUNT                 000007AD R      03
ARG_LIST                  0000081C R      03
ASTPAR_ERRMSG             00000A77 R      02
BADLEN                    0000022A R      02
BEGIN_MSGM              = 00000008
BEGIN_MSGV              = 00000003
BRDCST_BUF                000008F8 R      03
BRDCST_BUF_LEN          = 0000002E
BRDCST_NEDEDM           = 00000040
BRDCST_NEDEDV           = 00000006
BUFFER                    0000005B R      03
BUFFER_PTR                00000053 R      03
CAB1_UNPLUGED             000003FD R      02
CAB2_UNPLUGED             000003AA R      02
CASE_FAILED               0000029A R      02
CCASTHAND                 00001548 R      05
CHECK_IOSB                00001057 R      05
CHF$L_SIGARGLST         = 00000004
CHF$L_SIG_ARG1          = 00000008
CHF$L_SIG_ARGS          = 00000000
CHF$L_SIG_NAME          = 00000004
CHK_LOOP                  000012EA R      05
CHK_QIO_AST               00001070 R      05
CLEAN_EXIT                00000F37 R      05
CNTRLCMSG                 000002CF R      02
CNTR_TBL                  000C088D R      02
COMMON                    0000150F R      05
CONTROLLER                00000031 R      02
CONT_DESC                 000009DE R      02
COUNTER                   0000CBDE R      03
COUNTER_CHECK             000012D1 R      05
COUNTER_EXIT              000013D0 R      05
COUNTER_MSG               00000271 R      02
CS1                       000000A1 R      02
CS3                       000000B3 R      02
CVTRSL                    00000814 R      03
DDB_RFA                   00000E84 R      03
DEAD_CTRLNAME             00000310 R      02
DEV$V_TRM                 ********    X   05
DEVDEP_SIZE             = 00000000
DEVDSC                    000006BB R      03
DEVNAM_LEN                00000787 R      03
DEV_NAME                  000006DA R      03
DIAG_BUF                  00000C00 R      03
DIAG_BUF_SIZE           = 00000050
DIB                       000006E9 R      03
DIB$B_DEVCLASS          = 00000004
DIB$B_DEVTYPE           = 00000005
DIB$K_LENGTH            = 00000074
DIBBUF                    000006F1 R      03
DO_BRDCST                 000009A8 R      05
DUMMY_FAB                 00000EDC R      03
DUMMY_RAB                 00000F2C R      03
DUMP                      000000E7 R      02
DUMP_MODEM              = 00000001
DUMP_MODEV              = 00000000
DUMP_SIGNAL               000012BD R      05
DUMP_STAMPS               00001256 R      05
DVI$_DEVNAM             = 00000020
ECHO_SWITCH               0000086E R      03
EFN2                    = 00000004
END_ADR                   00000BE2 R      03
END_UPDATE                000016F4 R      05
ERRCNT_BUF                000009DE R      03
ERRCNT_LEN              = 00000080
ERRCOUNT_DESC             000009D6 R      03
ERROR_COUNT               00000765 R      03
ERROR_EXIT                0000157F R      05
ERROR_TEST                00000DAF R      05
ERRTEST_MSG               00000A5B R      02
ERRTST_ERR                00000E51 R      05
ERRTST_P2BUF              000009CE R      03
ERRTST_P2DESC             000009C6 R      03
ERRTST_P2LEN            = 00000008
EXIT_DESC                 0000079D R      03
EXIT_HANDLER              000015FF R      05
FAB$B_BID               = 00000000
FAB$B_FNS               = 00000034
FAB$C_BID               = 00000003
FAB$C_BLN               = 00000050
FAB$C_SEQ               = 00000000
FAB$C_VAR               = 00000002
FAB$L_ALQ               = 00000010
FAB$L_DEV               = 00000040
FAB$L_FNA               = 0000002C
FAB$L_FOP               = 00000004
FAB$L_STS               = 00000008
FAB$L_STV               = 0000000C
FAB$V_CHAN_MODE         = 00000002
FAB$V_CR                = 00000001
FAB$V_FILE_MODE         = 00000004
FAB$V_GET               = 00000001
FAB$V_LNM_MODE          = 00000000
FAB$V_PUT               = 00000000
FAB$V_UFO               = 00000011
FAB$V_UPD               = 00000003
FAB$V_UPI               = 00000006
FAB$W_GBC               = 00000048
```

FAIL_OUT		00000FE6	R	05	MSG_BLOCK		00000799	R	03
FAO_BUF		0000004B	R	03	NAME_LEN	=	0000000F		
FILE		000009E6	R	02	NEW_NODE		000007C0	R	03
FIND_IT		000001E9	R	05	NIADRTESTING		000U013C	R	02
FLAG		00000049	R	03	NIADRWRONG		00000CF3	R	02
FLAG_SHUTDNM	=	00000010			NMA$C_CTLIN_BSM	=	000003F7		
FLAG_SHUTDNV	=	00000004			NMA$C_CTLIN_CDC	=	00000425		
FORWARD_MSG_SIZ	=	0000000E			NMA$C_CTLIN_LBE	=	00000411		
FOUND_IT		00000285	R	05	NMA$C_CTLIN_OVR	=	00000428		
GETSYI_ITMLST		00000AD1	R	02	NMA$C_CTLIN_RFL	=	00000426		
HALFMIN		000009BE	R	02	NMA$C_CTLIN_SBU	=	00000429		
HARD_ADR	=	0000002A			NMA$C_CTLIN_UBU	=	0000042A		
ID_CTRSTR		00000180	R	02	NMA$C_LINCN_NOR	=	00000000		
ID_FNDM	=	00000020			NMA$C_LINMC_SET	=	00000001		
ID_FNDV	=	00000005			NMA$C_PCLI_BFN	=	00000451		
ILLEGAL_REC		000003DA	R	02	NMA$C_PCLI_BUS	=	00000AF1		
INADDRESS		00000775	R	03	NMA$C_PCLI_CON	=	00000456		
INIDEV_UPDERR		0000C0C2	R	02	NMA$C_PCLI_CRC	=	00000B1C		
INI_FAB		00000DF0	R	03	NMA$C_PCLI_DCH	=	00000B1B		
INI_RAB		00000E40	R	03	NMA$C_PCLI_EKO	=	00000B1F		
INPUT_ITMLST		00000083	R	02	NMA$C_PCLI_MCA	=	00000B0F		
IO$M_CLR_COUNT		********	X	05	NMA$C_PCLI_MLT	=	00000B19		
IO$M_CTRL		********	X	05	NMA$C_PCLI_PAD	=	00000B1A		
IO$M_CTRLCAST		********	X	05	NMA$C_PCLI_PRM	=	00000B18		
IO$M_NOW		********	X	05	NMA$C_PCLI_PTY	=	00000B0E		
IO$M_RD_COUNT		********	X	05	NMA$C_STATE_OFF	=	00000001		
IO$M_SHUTDOWN		********	X	05	NMA$C_STATE_ON	=	00000000		
IO$M_STARTUP		********	X	05	NMA$S_CNT_TYP	=	0000000C		
IO$_READLBLK		********	X	05	NMA$S_CNT_WID	=	00000002		
IO$_READVBLK		********	X	05	NMA$V_CNT_MAP	=	0000000C		
IO$_SENSEMODE		********	X	05	NMA$V_CNT_TYP	=	00000000		
IO$_SETCHAR		********	X	05	NMA$V_CNT_WID	=	0000000D		
IO$_SETMODE		********	X	05	NOUNIT_SELECTED		00000357	R	02
IO$_WRITELBLK		********	X	05	NO_CTRLNAME		000002F0	R	02
IO$_WRITEVBLK		********	X	05	NO_RMS_AST_TABLE		0000005E	R	02
ITERATION		00000791	R	03	NRAT_LENGTH	=	00000014		
LC_BITM	=	00000020			OTS$CVT_TI_L		********	X	05
LIB$CVT_HTB		********	X	05	OUTADDRESS		0000077D	R	03
LIB$SIGNAL		********	X	05	OUT_OF_SEQ_MSG		0000075B	R	02
LIN	=	00000000			P1BUF		00000834	R	03
LOGNAM		0000004D	R	02	P2_DESC		00000884	R	03
LONGORSHRTM	=	00000080			P2_DESC1		000008D4	R	03
LONGORSHRTV	=	00000007			P2_PARAM		0000083C	R	03
LOOPBACK_ACK		00C00C54	R	03	P2_PARAM1		0000088C	R	03
LOOPBACK_AST		000010F1	R	05	PAD_SWITCH		00000862	R	03
LOOPBACK_FAIL_MSG		00000667	R	02	PAGES	=	00000007		
LOOPBACK_TMO		00001087	R	05	PASS		00000795	R	03
LOOPBACK_TMO_MSG		00000723	R	02	PASS_MSG		0000085A	R	02
LOOP_PROTOCOL	=	00000090			PC1...	=	000008B7	R	02
MAXBUF		00000C50	R	03	PC2...	=	0000098E	R	02
MAXBUF_MSG		00000AE1	R	02	PCSR1	=	00000001A		
MAX_DEV_DESIG	=	0000000A			PCSR1$S_SLFTST	=	00000006		
MAX_PROC_NAME	=	0000000F			PCSR1$V_ICAB	=	0000000E		
MAX_UNABUF_SIZE	=	000005DC			PCSR1$V_SLFTST	=	00000008		
MAX_UNIT_DESIG	=	00000005			PCSR1$V_XPWR	=	0000000F		
MIN_MAXBUF	=	00000630			PMTSIZ	=	00000019		
MODE		00000041	R	02	PRM	=	00000064		

```
PROCESS_NAME                    000006C3 R      03
PROCESS_NAME_FREE             = 0000000B
PROC_CONT_NAME                  00000093 R      05
PROMPT                          00000A21 R      02
PROTOCOL                        00000868 R      03
QUAD_STATUS                     0000076D R      03
RAB$B_PSZ                     = 00000034
RAB$B_RAC                     = 0000001E
RAB$C_BID                     = 00000001
RAB$C_BLN                     = 00000044
RAB$C_RFA                     = 00000002
RAB$C_SEQ                     = 00000000
RAB$L_CTX                     = 00000018
RAB$L_FAB                     = 0000003C
RAB$L_PBF                     = 00000030
RAB$L_ROP                     = 00000004
RAB$L_STS                     = 00000008
RAB$L_STV                     = 0000000C
RAB$V_PMT                     = 0000001E
RAB$W_RFA                     = 00000010
RAB$W_RSZ                     = 00000022
RANDOM1                         00000789 R      03
RANDOM2                         0000078D R      03
RCV_IOSB                        00000BEE R      03
RDID_EFN                      = 00000007
READ_ERRCOUNT                   00000EC7 R      05
READ_SIZE                     = 000005DC
RECORD                          000009F2 R      02
RECVPOOL_SIZ                  = 00000004
RECV_AST                        00001182 R      05
RECV_BUF_ADR                    00000BFC R      03
RECV_EFN                      = 00000006
RECV_STAMP                      00001221 R      05
REC_SIZE                      = 00000028
REMOTE_ID_CTR                   000001BE R      02
REMOTE_LOOPBACK_TEST            00000BC2 R      05
REMOTE_NOTFND                   00000200 R      02
REPORT                          00000093 R      02
REQUEST_BUF                     000000DF R      03
REQUEST_BUFSIZ                = 000005DC
REQUEST_FND                     00000E58 R      05
REQUEST_NOTFND                  00000EBD R      05
RESTART                         0000085A R      05
RETRANS_MSG                     000006E1 R      02
RETRY_COUNT                   = 00000004
RETRY_MSG                       000005E9 R      02
RMS$_BLN                        ******** X      02
RMS$_BUSY                       ******** X      02
RMS$_CDA                        ******** X      02
RMS$_FAB                        ******** X      02
RMS$_FACILITY                 = 00000001
RMS$_RAB                        ******** X      02
RMS_ERROR                       000014D9 R      05
RMS_ERR_STRING                  00000A00 R      02
RSLBUF                          000007CC R      03
RSLLEN                          000007C8 R      03
RW_TIME_ID                    = 00000003
SAFE_TO_UPDM                  = 00000004
SAFE_TO_UPDV                  = 00000002
SAVE_STAT                       00000BF6 R      03
SBP                             00000C58 R      03
SEC$M_EXPREG                    ********   X    05
SEC$M_GBL                       ********   X    05
SELF_ADR                        000008FC R      03
SENSE_ERRMSG                    00000A3A R      02
SENSE_P1BUF                     00000926 R      03
SENSE_P2BUF                     00000936 R      03
SENSE_P2DESC                    0000092E R      03
SENSE_P2LEN                   = 00000090
SENSE_TEST                      00000D3C R      05
SHR$_ABENDD                   = 000010E0
SHR$_BEGIND                   = 00001038
SHR$_ENDEDD                   = 00001080
SHR$_TEXT                     = 00001130
SIZE_TBL                        0000098E R      02
SIZE_TBL1                       0000099E R      02
SIZE_TBL_SIZE                 = 00000004
SLFTST_FAILED                   00000A9A R      02
SS$_ABORT                       ********   X    05
SS$_BADPARAM                    ********   X    05
SS$_BUFFEROVF                   ********   X    05
SS$_CONTROLC                    ********   X    05
SS$_DATAOVERUN                  ********   X    05
SS$_ENDOFFILE                   ********   X    05
SS$_IVBUFLEN                    ********   X    05
SS$_NORMAL                      ********   X    05
SS$_NOSUCHSEC                   ********   X    05
SS$_NOTRAN                      ********   X    05
SS$_SSFAIL                      ********   X    05
SS$_TIMEOUT                     ********   X    05
SS$_WASSET                      ********   X    05
SSERROR                         000013F6 R      05
SS_SYNCH_EFN                  = 00000003
STAMP_BUFFER                    00000C5C R      03
START_CONT                      000007DA R      05
STATUS                          00000769 R      03
STR$UPCASE                      ********   X    05
STS$K_ERROR                   = 00000002
STS$K_INFO                    = 00000003
STS$K_SEVERE                  = 00000004
STS$K_SUCCESS                 = 00000001
STS$K_WARNING                 = 00000000
STS$M_INHIB_MSG               = 10000000
STS$S_FAC_NO                  = 0000000C
STS$S_SEVERITY                = 00000003
STS$V_FAC_NO                  = 00000010
STS$V_SEVERITY                = 00000000
SUC_EXIT                        00000F60 R      05
SUPDEV_GBLSEC                   00000020 R      02
SUP_FAB                         00000E8C R      03
SYI$_MAXBUF                   = 0000104F
SYS$ASSIGN                      ********  GX    05
SYS$CANCEL                      ********  GX    05
SYS$CANTIM                      ********  GX    05
```

```
SYS$CLREF                ********  GX    05        TIME_SUC_OUT                      000013EE R      05
SYS$CONNECT              ********  GX    05        TSCLC                           =  00000100
SYS$CRMPSC               ********  GX    05        TTCHAN                            00000000 R      03
SYS$DASSGN               ********  GX    05        TTNAME                            0000000A R      03
SYS$DCLEXH               ********  GX    05        TTNAME_LEN                      =  00000009
SYS$EXIT                 ********  GX    05        TTNAME_ROPTR                      000002C7 R      02
SYS$EXPREG               ********  GX    05        TTNAME_RWPTR                      00000002 R      03
SYS$FAO                  ********   X    05        TWENSEC                           000009B6 R      02
SYS$FAOL                 ********  GX    05        TWOMIN                            000009C6 R      02
SYS$GET                  ********  GX    05        UETP                            =  00740000
SYS$GETDEV               ********  GX    05        UETP$_ABENDD                    =  007410E0
SYS$GETDVI               ********  GX    05        UETP$_ABORTC                    =  0074832B
SYS$GETMSG               ********  GX    05        UETP$_BEGIND                    =  00741038
SYS$GETSYI               ********  GX    05        UETP$_DATADEVERR                =  00748018
SYS$GETTIM               ********  GX    05        UETP$_DENOSU                    =  00748333
SYS$INPUT                00000072 R      02        UETP$_DEUNUS                    =  0074819A
SYS$MGBLSC               ********  GX    05        UETP$_ENDEDD                    =  00741080
SYS$OPEN                 ********  GX    05        UETP$_ERBOXPROC                 =  00748020
SYS$PUTMSG               ********  GX    05        UETP$_FACILITY                  =  00000074
SYS$QIO                  ********  GX    05        UETP$_TEXT                      =  00741130
SYS$QIOW                 ********  GX    05        UETUNAS00                         00000000 RG     05
SYS$READEF               ********  GX    05        UETUNT$B_FLAGS                  =  00000008
SYS$SETAST               ********  GX    05        UETUNT$B_TYPE                   =  00000008
SYS$SETEF                ********  GX    05        UETUNT$C_FAB                    =  00000110
SYS$SETIMR               ********  GX    05        UETUNT$C_INDSIZ                 =  000001A4
SYS$SETPRN               ********  GX    05        UETUNT$K_DEVDEP                 =  000001A4
SYS$SETSFM               ********  GX    05        UETUNT$K_FAB                    =  C0000110
SYS$TRNLOG               ********  GX    05        UETUNT$K_RAB                    =  00000160
SYS$UPDATE               ********  GX    05        UETUNT$M_TESTABLE               =  00000002
SYS$WAITFR               ********  GX    05        UETUNT$T_FILSPC                 =  00000014
SYSIN_FAB                00000D5C R      03        UETUNT$V_TESTABLE               =  00000001
SYSIN_RAB                00000DAC R      03        UETUNT$W_SIZE                   =  00000009
TBL_END                  000008B7 R      02        UNA_SHUTDOWN                      0000102C R      05
TBL_SIZE                 =  00000007               UNA_STARTUP                       00000FF8 R      05
TENSEC                   000009AE R      02        UNIT_DESC                         000009D6 R      02
TEST_NAME                0000000F R      02        UNIT_LIST                         000007B8 R      03
TEST_OVERM               =  00000002               UNIT_LOOP                         00001671 R      05
TEST_OVERV               =  00000001               UNIT_NUMBER                       00000785 R      03
TEXT_BUFFER              =  00000084               UPDATE_FAILED                     000016D6 R      05
THREEMIN                 000009CE R      02        WRITE_SIZE                      =  000005DC
TIMEOUT1                 0000040E R      02        XE_CHAN                           00000818 R      03
TIMEOUT10                000005B1 R      02        XE_CHAN1                          0000081A R      03
TIMEOUT11                000005CB R      02        XE_IOSB                           00000BE6 R      03
TIMEOUT2                 0000042F R      02        XM$M_CHR_LOOPB                  =  00000002
TIMEOUT3                 00000453 R      02        XMIT                              00000878 R      05
TIMEOUT4                 00000483 R      02        XMIT_BUF_ADR                      00000BF8 R      03
TIMEOUT5                 000004A6 R      02        XMIT_EFN                        =  00000005
TIMEOUT6                 000004DC R      02        XMIT_LOOP                         0000085A R      05
TIMEOUT7                 0000050A R      02        XMIT_SETUP                        00000BAF R      05
TIMEOUT8                 00000541 R      02        XMIT_STAMP                        0000121B R      05
TIMEOUT9                 0000056B R      02        XMIT_START                        00000951 R      05
TIME_ERR_OUT             000013D1 R      05
TIME_ID_1                =  00000001
TIME_ID_2                =  00000002
TIME_IT                  00000689 R      05
TIME_STAMP_INTRO_MSG     0000081D R      02
TIME_STAMP_MSG           000007DB R      02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	00000B47 (2887.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RWDATA	00000F70 (3952.)	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
UNAS	0000172A (5930.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.07	00:00:00.34
Command processing	145	00:00:00.68	00:00:03.89
Pass 1	1212	00:00:37.89	00:01:19.61
Symbol table sort	0	00:00:03.65	00:00:07.76
Pass 2	822	00:00:10.43	00:00:22.13
Symbol table output	0	00:00:00.36	00:00:00.66
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	2221	00:00:53.12	00:01:54.44

The working set limit was 2000 pages.
215843 bytes (422 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2485 non-local and 94 local symbols.
2987 source lines were read in Pass 1, producing 59 object records in Pass 2.
69 pages of virtual memory were used to define 62 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[UETP.OBJ]UETP.MLB;1	2
_255DUA28:[SYS.OBJ]LIB.MLB;1	1
_255DUA28:[SYSLIB]STARLET.MLB;2	55
TOTALS (all libraries)	58

2722 GETS were required to define 58 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:UETUNAS00/OBJ=OBJ$:UETUNAS00 MSRC$:UETUNAS00/UPDATE=(ENH$:UETUNAS00)+EXECML$/LIB+LIB$:UETP/LIB
```