```
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**UETDMPF00

```
UU      UU  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    MM      MM  PPPPPPPP    FFFFFFFFFF    000000      000000
UU      UU  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    MM      MM  PPPPPPPP    FFFFFFFFFF    000000      000000
UU      UU  EE              TT      DD      DD  MMMM  MMMM  PP      PP  FF           00      00  00      00
UU      UU  EE              TT      DD      DD  MMMM  MMMM  PP      PP  FF           00      00  00      00
UU      UU  EE              TT      DD      DD  MM  MM  MM  PP      PP  FF           00    0000  00    0000
UU      UU  EE              TT      DD      DD  MM  MM  MM  PP      PP  FF           00    0000  00    0000
UU      UU  EEEEEEEE        TT      DD      DD  MM      MM  PPPPPPPP    FFFFFFFF     00  00  00  00  00  00
UU      UU  EEEEEEEE        TT      DD      DD  MM      MM  PPPPPPPP    FFFFFFFF     00  00  00  00  00  00
UU      UU  EE              TT      DD      DD  MM      MM  PP          FF           0000    00  0000    00
UU      UU  EE              TT      DD      DD  MM      MM  PP          FF           0000    00  0000    00
UU      UU  EE              TT      DD      DD  MM      MM  PP          FF           00      00  00      00      ....
UU      UU  EE              TT      DD      DD  MM      MM  PP          FF           00      00  00      00      ....
UUUUUUUUUU  EEEEEEEEEE      TT      DDDDDDDD    MM      MM  PP          FF             000000      000000        ....
UUUUUUUUUU  EEEEEEEEEE      TT      DDDDDDDD    MM      MM  PP          FF             000000      000000        ....

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

Table of contents

```
0000     1          .TITLE UETDMPF00 VAX/VMS UETP DEVICE TEST FOR DMP 11/ DMF-32 Sync Line
0000     2          .IDENT  'V04-001'
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    18 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27
0000    28 ;++
0000    29 ; FACILITY:
0000    30 ;       This module will be distributed with VAX/VMS under the [SYSTEST]
0000    31 ;       account.
0000    32 ;
0000    33 ; ABSTRACT:
0000    34 ;       This is the test program for DMP 11 / DMF 32 Sync line device test
0000    35 ;
0000    36 ; ENVIRONMENT:
0000    37 ;       This program will run in user access mode, with AST enabled except
0000    38 ;       during error processing. This program requires the following
0000    39 ;       privileges and  quotas:  none.
0000    40 ;
0000    41 ;--
0000    42 ;
0000    43 ; AUTHOR: Paul Jeng,    CREATION DATE: Sep, 1981
0000    44 ;
0000    45 ; MODIFIED BY:
0000    46 ;
0000    47 ;       V04-001 RNH0009         Richard N. Holstein,    07-Sep-1984
0000    48 ;               Remove entirely the forced error for too big a buffer, since
0000    49 ;               either SS$_BADPARAM or SS$_EXQUOTA could be returned depending
0000    50 ;               on SYSGEN parameters.
0000    51 ;
0000    52 ;       V03-010 RNH0008         Richard N. Holstein,    07-Apr-1984
0000    53 ;               Adapt to driver fix which allocated write buffers dynamically -
0000    54 ;               we can't force an SS$_BADPARAM unless we exceed absolute max.
0000    55 ;
0000    56 ;       V03-009 RNH0007         Richard N. Holstein,    15-Feb-1984
0000    57 ;               Take advantage of the new UETP message codes.  Fix SSERROR
```

```
0000     58 ;                  interaction with RMS_ERROR.
0000     59 ;
0000     60 ;        V03-008 RNH0006         Richard N  Holstein,    19-Dec-1983
0000     61 ;                Give correct sentinels to Test Controller.
0000     62 ;
0000     63 ;        V03-007 RNH0005         Richard N. Holstein,    11-Nov-1983
0000     64 ;                Use decimal conversion routine for unit numbers.
0000     65 ;
0000     66 ;        V03-006 RNH0004         Richard N. Holstein,    29-Jun-1983
0000     67 ;                Rework error messages and error processing.
0000     68 ;
0000     69 ;        V03-005 RNH0003         Richard N. Holstein,    11-Mar-1983
0000     70 ;                Don't signal ending message in EXIT_HANDLER.
0000     71 ;
0000     72 ;        V03-004 RNH0002         Richard N. Holstein,    01-Mar-1983
0000     73 ;                Fix ERROR_COUNT bug.
0000     74 ;
0000     75 ;        V03-003 LDJ0002         Larry D. Jones,         10-Feb-1983
0000     76 ;                Allow for longer device names.
0000     77 ;
0000     78 ;        V03-002 LDJ0001         Larry D. Jones,         06-Nov-1982
0000     79 ;                Fixed a loop mode assign channel bug.
0000     80 ;
0000     81 ;        V03-001 RNH0001         Richard N. Holstein,    15-Oct-1982
0000     82 ;                Miscellaneous fixes listed in the V3B UETP Workplan.
0000     83 ;
0000     84 ;**
```

```
                     0000     86          .SBTTL  Declarations
                     0000     87 ;
                     0000     88 ; INCLUDE FILES:
                     0000     89 ;
                     0000     90 ;       SYS$LIBRARY:LIB.MLB     for general definitions
                     0000     91 ;       SHRLIB$:UETP.MLB        for UETP definitions
                     0000     92 ;
                     0000     93 ;
                     0000     94 ; MACROS:
                     0000     95 ;
                     0000     96          $CHFDEF                         ; Condition handler frame definitions
                     0000     97          $DEVDEF                         ; Device definitions
                     0000     98          $DIBDEF                         ; Device Information Block
                     0000     99          $DVIDEF                         ; $GETDVI ITMLST item codes
                     0000    100          $SHRDEF                         ; Shared messages
                     0000    101          $SSDEF                          ; System Service status codes
                     0000    102          $STSDEF                         ; Status return
                     0000    103          $UETUNTDEF                      ; UETP unit block offset definitions
                     0000    104          $UETPDEF                        ; UETP
                     0000    105          $XMDEF                          ; XMDRIVER symbols
                     0000    106          $NMADEF                         ; Network management definition
                     0000    107 ;
                     0000    108 ; EQUATED SYMBOLS:
                     0000    109 ;
                     0000    110 ;   Facility number definitions:
00000001             0000    111          RMS$_FACILITY = 1
                     0000    112
                     0000    113 ;   SHR message definitions:
00740000             0000    114          UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
007410E0             0000    115          UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
00741038             0000    116          UETP$_BEGIND = UETP!SHR$_BEGIND
00741080             0000    117          UETP$_ENDEDD = UETP!SHR$_ENDEDD
00741098             0000    118          UETP$_OPENIN = UETP!SHR$_OPENIN
00741130             0000    119          UETP$_TEXT   = UETP!SHR$_TEXT
                     0000    120
                     0000    121 ;   Internal flag bits...:
00000001             0000    122          TEST_OVERV   = 1                ; Set when test is over
00000002             0000    123          SAFE_TO_UPDV = 2                ; Set if it's safe to update UETINIDEV
00000003             0000    124          BEGIN_MSGV   = 3                ; Set if "BEGIN" msg has been printed
00000004             0000    125          MODE_IS_ONEV = 4                ; Set when the MODE is ONE
00000005             0000    126          FLAG_SHUTDNV = 5                ; Set to indicate device should be
                     0000    127                                          ; shutdown if errors occur
                     0000    128 ;   ...and corresponding masks:
00000002             0000    129          TEST_OVERM   = 1@TEST_OVERV
00000004             0000    130          SAFE_TO_UPDM = 1@SAFE_TO_UPDV
00000008             0000    131          BEGIN_MSGM   = 1@BEGIN_MSGV
00000010             0000    132          MODE_IS_ONEM = 1@MODE_IS_ONEV
00000020             0000    133          FLAG_SHUTDNM = 1@FLAG_SHUTDNV
                     0000    134
                     0000    135 ;   Miscellany:
00000020             0000    136          LC_BITM      = ^X20             ; Mask to convert lower case to upper
00000028             0000    137          REC_SIZE     = 40               ; UETINIDEV.DAT record size
000000FA             0000    138          TEXT_BUFFER  = 250              ; Internal text buffer size
00000004             0000    139          EFN2         = 4                ; EFN used for three minute timer
00000003             0000    140          SS_SYNCH_EFN = 3                ; Synch miscellaneous system services
0000000F             0000    141          MAX_PROC_NAME = 15              ; Longest possible process name
0000000A             0000    142          MAX_DEV_DESIG = 10              ; Longest possible controller name
```

```
00000005  0000   143          MAX_UNIT_DESIG= 5                ; Longest possible unit number
00000200  0000   144          MAX_MSG_LEN   = 512              ; maximum message length
00000001  0000   145          TIME_ID_1     = 1                ; Timer id to prevent hung
00000003  0000   146          RW_TIME_ID    = 3                ; Timer to prevent hung when Read/write
00000010  0000   147          LIMIT         = 16               ; Loop count for each message length
00000008  0000   148          RECV_EFN      = 8
00000005  0000   149          XMIT_EFN      = 5                ; EFN for QIO write
00000064  0000   150          PRM           = 100              ; AST parameter for test
00000000  0000   151          DEVDEP_SIZE   = 0                ; Size of device dependent part of UETUNT
00000000  0000   152          WRITE_SIZE    = 0                ; Size of device write buffer
00000000  0000   153          READ_SIZE     = 0                ; Size of device read buffer
          0000   154
          0000   155          PAGES = <<UETUNT$C_INDSIZ+-      ; Add together all of the pieces...
          0000   156                    DEVDEP_SIZE+-          ; ...which make up a UETP unit block...
          0000   157                    WRITE_SIZE+-           ; ...to give to the $EXPREG service below
          0000   158                    READ_SIZE+-
00000001  0000   159                    511>7512>
          0000   160
0000001B  0000   161          ESC = ^X1B                       ; ESC character
00000004  0000   162          RECVPOOL_SIZ = 4                 ; Number of preallocated message block
```

```
                                0000   164          .SBTTL  Read-Only Data
                                0000   165          .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                0000   166
                                0000   167 ACNT_NAME:                                ; Process name on exit
53 45 54 53 59 53 00000008'010E0000'   168          .ASCID  /SYSTEST/
                                   54  000E
                                000F   169
                                000F   170 TEST_NAME:                                ; This test name
50 4D 44 54 45 55 00000017'010E0000'   171          .ASCID  /UETDMPF00/
                             30 30 46  001D
                                0020   172
                                0020   173 SUPDEV_GBLSEC:                            ; How we access UETSUPDEV.DAT
50 55 53 54 45 55 00000028'010E0000'   174          .ASCID  /UETSUPDEV/
                             56 45 44  002E
                                0031   175
                                0031   176 CONTROLLER:                               ; Logical name of controller
41 4E 4C 52 54 43 00000039'010E0000'   177          .ASCID  /CTRLNAME/
                                45 4D  003F
                                0041   178
                                0041   179 MODE:                                     ; Run mode logical name
      45 44 4F 4D 00000049'010E0000'   180          .ASCID  /MODE/
                                004D   181
                                004D   182 NO_RMS_AST_TABLE:                         ; List of errors for which...
                          00000000'    183          .LONG   RMS$_BLN                ; ...RMS cannot deliver an AST...
                          00000000'    184          .LONG   RMS$_BUSY               ; ...even if one has an ERR= arg
                          00000000'    185          .LONG   RMS$_CDA                ; Note that we can search table...
                          00000000'    186          .LONG   RMS$_FAB                ; ...via MATCHC since <31:16>...
                          00000000'    187          .LONG   RMS$_RAB                ; ...pattern can't be in <15:0>
                          00000014     188 NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                0061   189
                                0061   190 SYS$INPUT:                                ; Name of device from which...
4E 49 24 53 59 53 00000069'010E0000'   191          .ASCID  /SYS$INPUT/             ; ...the test can be aborted
                             54 55 50  006F
                                0072   192
                                0072   193 INPUT_ITMLST:                            ; $GETDVI arg list for SYS$INPUT
                           0020 0040   194          .WORD   64,DVI$_DEVNAM          ; We need the equivalence name
                  0000000C'00000014'   195          .LONG   BUFFER,BUFFER_PTR
                           00000000    196          .LONG   0                       ; Terminate the list
                                0082   197
                                0082   198 CS1:                                     ; Device class and type control string
21 20 42 58 32 21 0000008A'010E0000'   199          .ASCID  /!2XB !2XB /
                             20 42 58 32  0090
                                0094   200
                                0094   201 CS3:                                     ; Device class-only control string
2A 20 42 58 32 21 0000009C'010E0000'   202          .ASCID  /!2XB **/
                                   2A  00A2
                                00A3   203
                                00A3   204 CNTRLCMSG:
65 74 72 6F 62 41 000000AB'010E0000'   205          .ASCID  \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64  00B1
                   43 2F 4C 52 54 43 20  00BD
                                00C4   206
                                00C4   207 NO_CTRLNAME:
6E 6F 63 20 6F 4E 000000CC'010E0000'   208          .ASCID  /No controller specified./
63 65 70 73 20 72 65 6C 6C 6F 72 74  00D2
                   2E 64 65 69 66 69  00DE
                                00E4   209
```

```
                                                00E4    210 DEAD_CTRLNAME:
20 74 27 6E 61 43 000000EC'010E0000'            00E4    211         .ASCID  /Can't test controller .AS, marked as unusable in UETINIDEV.DAT./
6C 6F 72 74 6E 6F 63 20 74 73 65 74             00F2
72 61 6D 20 2C 53 41 21 20 72 65 6C             00FE
61 73 75 6E 75 20 73 61 20 64 65 6B             010A
4E 49 54 45 55 20 6E 69 20 65 6C 62             0116
          2E 54 41 44 2E 56 45 44 49            0122
                                                012B    212
                                                012B    213 NOUNIT_SELECTED:
69 6E 75 20 6F 4E 00000133'010E0000'            012B    214         .ASCID  /No units selected for testing./
20 64 65 74 63 65 6C 65 73 20 73 74             0139
2E 67 6E 69 74 73 65 74 20 72 6F 66             0145
                                                0151    215
                                                0151    216 ILLEGAL_REC:
61 67 65 6C 6C 49 00000159'010E0000'            0151    217         .ASCID  /Illegal record format in file UETINIDEV.DAT!/
72 6F 66 20 64 72 6F 63 65 72 20 6C             015F
20 65 6C 69 66 20 6E 69 20 74 61 6D             016B
41 44 2E 56 45 44 49 4E 49 54 45 55             0177
                              21 54             0183
                                                0185    218
                                                0185    219 PASS_MSG:
66 6F 20 64 6E 45 0000018D'010E0000'            0185    220         .ASCID  /End of pass !UL with !UL iterations at !%D./
69 77 20 4C 55 21 20 73 73 61 70 20             0193
61 72 65 74 69 20 4C 55 21 20 68 74             019F
44 25 21 20 74 61 20 73 6E 6F 69 74             01AB
                              2E                01B7
                                                01B8    221
                                                01B8    222 INIDEV_UPDERR:                          ; Error during exit handler
20 72 6F 72 72 45 000001C0'010E0000'            01B8    223         .ASCID  /Error updating UETINIDEV.DAT./
54 45 55 20 67 6E 69 74 61 64 70 75             01C6
   2E 54 41 44 2E 56 45 44 49 4E 49             01D2
                                                01DD    224
                                                01DD    225 THREEMIN:                               ; 3 minute delta time
                  FFFFFFFF 94B62E00             01DD    226         .LONG   -10*1000*1000*180,-1
                                                01E5    227
                                                01E5    228 HALFMIN:                                ; 30 seconds delta time
                  FFFFFFFF EE1E5D00             01E5    229         .LONG   -10*1000*1000*30,-1
                                                01ED    230
                                                01ED    231 UNIT_DESC:                              ; Descriptor used to convert unit #
                           00000005             01ED    232         .LONG   5
                           0000001A'            01F1    233         .ADDRESS BUFFER+6
                                                01F5    234
                                                01F5    235 CONT_DESC:                              ; Descriptor used to convert controller...
                       0000 0028                01F5    236         .WORD   REC_SIZE,0              ; ...from lowercase to uppercase
                           00000014'            01F9    237         .ADDRESS BUFFER
                                                01FD    238
                                                01FD    239 FILE:                                   ; Fills in RMS_ERR_STRING
      65 6C 69 66 00000205'010E0000'            01FD    240         .ASCID  /file/
                                                0209    241
                                                0209    242 RECORD:                                 ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 00000211'010E0000'            0209    243         .ASCID  /record/
                                                0217    244
                                                0217    245 RMS_ERR_STRING:                         ; Announces an RMS error
41 21 20 53 4D 52 0000021F'010E0000'            0217    246         .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53             0225
                  44 41 21 20 65 6C 69          0231
                                                0238    247
```

```
                                        0238   248 PROMPT:
64 20 72 65 6C 6C 0F 72 74 6E 6F 43     0238   249         .ASCII  /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65     0244
                              20        0250
                        00000019        0251   250         PMTSIZ = .-PROMPT
                                        0251   251
                                        0251   252 START_TO_MSG:
75 6F 65 6D 69 54 00000259'010E0000'    0251   253         .ASCID  /Timeout trying to start !AS./
20 6F 74 20 67 6E 69 79 72 74 20 74     025F
      2E 53 41 21 20 74 72 61 74 73     026B
                                        0275   254
                                        0275   255 RW_TO_MSG:
75 6F 65 6D 69 54 0000027D'010E0000'    0275   256         .ASCID  /Timeout while reading or writing !AS./
64 61 65 72 20 65 6C 69 68 77 20 74     0283
69 74 69 72 77 20 72 6F 20 67 6E 69     028F
               2E 53 41 21 20 67 6E     029B
                                        02A2   257
                                        02A2   258 START_CONT_PRM:
20 72 6F 72 72 45 000002AA'010E0000'    02A2   259         .ASCID  /Error starting up !AS as a controller./
20 70 75 20 67 6E 69 74 72 61 74 73     02B0
6E 6F 63 20 61 20 73 61 20 53 41 21     02BC
            2E 72 65 6C 6C 6F 72 74     02C8
                                        02D0   260
                                        02D0   261 START_TRIB_PRM:
20 72 6F 72 72 45 000002D8'010E0000'    02D0   262         .ASCID  /Error starting up !AS as a tributary./
20 70 75 20 67 6E 69 74 72 61 74 73     02DE
69 72 74 20 61 20 73 61 20 53 41 21     02EA
               2E 79 72 61 74 75 62     02F6
                                        02FD   263
                                        02FD   264 WRITE_PRM:
20 72 6F 72 72 45 00000305'010E0000'    02FD   265         .ASCID  /Error writing to !AS./
21 20 6F 74 20 67 6E 69 74 69 72 77     030B
                              2E 53 41  0317
                                        031A   266
                                        031A   267 READ_PRM:
20 72 6F 72 72 45 00000322'010E0000'    031A   268         .ASCID  /Error reading from !AS./
6D 6F 72 66 20 67 6E 69 64 61 65 72     0328
                        2E 53 41 21 20  0334
                                        0339   269
                                        0339   270 SENSE_PRM:
20 72 6F 72 72 45 00000341'010E0000'    0339   271         .ASCID  /Error sensing characteristics of !AS./
72 61 68 63 20 67 6E 69 73 6E 65 73     0347
20 73 63 69 74 73 69 72 65 74 63 61     0353
               2E 53 41 21 20 66 6F     035F
                                        0366   272
                                        0366   273 SET_PRM:
20 72 6F 72 72 45 0000036E'010E0000     0366   274         .ASCID  /Error setting characteristics of !AS./
72 61 68 63 20 67 6E 69 74 74 65 73     0374
20 73 63 69 74 73 69 72 65 74 63 61     0380
               2E 53 41 21 20 66 6F     038C
                                        0393   275
                                        0393   276 DMF_IOSB_DUMP:
74 73 20 4F 2F 49 0000039B'010E0000'    0393   277         .ASCID  \I/O status block completion status: !XW, transfer size: !XW,\-
63 20 6B 63 6F 6C 62 20 73 75 74 61     03A1
74 73 20 6E 6F 69 74 65 6C 70 6D 6F     03AD
74 20 2C 57 58 21 20 3A 73 75 74 61     03B9
65 7A 69 73 20 72 65 66 73 6E 61 72     03C5
```

```
                                2C 57 58 21 20 3A   03D1
65 74 63 61 72 61 68 63 5F 21 2F 21   03D7    278                 \!/!_characteristics: !XB, status: .XB, error summary: !XB.\
42 58 21 20 3A 73 63 69 74 73 69 72   03E3
58 21 20 3A 73 75 74 61 74 73 20 2C   03EF
6D 75 73 20 72 6F 72 72 65 20 2C 42   03FB
      2E 42 58 21 20 3A 79 72 61 6D   0407
                                      0411    279
                                      0411    280 DMP_IOSB_DUMP:
74 73 20 4F 2F 49 00000419'010E0000'  0411    281         .ASCID  \I/O status block completion status: !XW, transfer size: !XW,\-
63 20 6B 63 6F 6C 62 20 73 75 74 61   041F
74 73 20 6E 6F 69 74 65 6C 70 6D 6F   042B
74 20 2C 57 58 21 20 3A 73 75 74 61   0437
65 7A 69 73 20 72 65 66 73 6E 61 72   0443
                   2C 57 58 21 20 3A  044F
65 74 63 61 72 61 68 63 5F 21 2F 21   0455    282                 \!/!_characteristics: !XB, status: !XB, error summary: !XB,\-
42 58 21 20 3A 73 63 69 74 73 69 72   0461
58 21 20 3A 73 75 74 61 74 73 20 2C   046D
6D 75 73 20 72 6F 72 72 65 20 2C 42   0479
      2C 42 58 21 20 3A 79 72 61 6D   0485
75 6E 20 6C 61 74 6F 74 5F 21 2F 21   048F    283                 \!/!_total number of errors: !XB.\
6F 72 72 65 20 66 6F 20 72 65 62 6D   049B
            2E 42 58 21 20 3A 73 72   04A7
                                      04AF    284
                                      04AF    285 COMP_STATUS_MSG:
72 75 6C 69 61 46 000004B7'010E0000'  04AF    286         .ASCID  /Failure during forced error tests,/<13><10><9>/expected: "/
72 6F 66 20 67 6E 69 72 75 64 20 65   04BD
65 74 20 72 6F 72 72 65 20 64 65 63   04C9
63 65 70 78 65 09 0A 0D 2C 73 74 73   04D5
                  22 20 3A 64 65 74   04E1
                                      04E7    287
                                      04E7    288 RECEIVED_MSG:
72 09 0A 0D 2C 22 000004EF'010E0000'  04E7    289         .ASCID  /",/<13><10><9>/received: "/
      22 20 3A 64 65 76 69 65 63 65   04F5
                                      04FF    290
                                      04FF    291 RECV_ERR_MSG:
76 69 65 63 65 52 00000507'010E0000'  04FF    292         .ASCID  /Received message error, good data is !XB, bad data is !XB /
65 20 65 67 61 73 73 65 6D 20 64 65   050D
64 20 64 6F 6F 67 20 2C 72 6F 72 72   0519
20 2C 42 58 21 20 73 69 20 61 74 61   0525
20 73 69 20 61 74 61 64 20 64 61 62   0531
                        20 42 58 21   053D
                                      0541    293
                                      0541    294 SENSE_ERRMSG:
20 72 6F 72 72 45 00000549'010E0000'  0541    295         .ASCID  \Error in sense mode test, extended characteristic parameter\-
64 6F 6D 20 65 73 6E 65 73 20 6E 69   054F
65 74 78 65 20 2C 74 73 65 74 20 65   055B
74 63 61 72 61 68 63 20 64 65 64 6E   0567
61 72 61 70 20 63 69 74 73 69 72 65   0573
                     72 65 74 65 6D   057F
68 74 69 77 20 57 58 21 5F 21 2F 21   0584    296                 \!/!_!XW with value !XL not matched by any of those returned.\
6E 20 4C 58 21 20 65 75 6C 61 76 20   0590
62 20 64 65 68 63 74 61 6D 20 74 6F   059C
6F 68 74 20 66 6F 20 79 6E 61 20 79   05A8
2E 64 65 6E 72 75 74 65 72 20 65 73   05B4
                                      05C0    297
                                      05C0    298 ERRTEST_MSG:
20 72 6F 72 72 45 000005C8'010E0000'  05C0    299         .ASCID  /Error in error test /
```

```
73 65 74 20 72 6F 72 72 65 20 6E 69  05CE
                              20 74  05DA
```

```
                     05DC   301          .SBTTL  Read/Write Data
                 00000000   302          .PSECT  RWDATA,WRT,NOEXE,PAGE
                     0000   303
                     0000   304 TTCHAN:                                  ; Channel associated with ctrl. term.
                0000 0000   305          .WORD   0
                     0002   306
                     0002   307 FLAG:                                    ; Miscellaneous flag bits
                0000 0002   308          .WORD   0                       ; (See Equated Symbols for definitions)
                     0004   309
                     0004   310 FAO_BUF:                                 ; FAO output string descriptor
           0000 00FA 0004   311          .WORD   TEXT_BUFFER,0
            00000014' 0008   312          .ADDRESS BUFFER
                     000C   313
                     000C   314 BUFFER_PTR:                              ; Fake .ASCID buffer for misc. strings
           0000 00FA 000C   315          .WORD   TEXT_BUFFER,0           ; A word for length, a word for desc.
            00000014' 0010   316          .ADDRESS BUFFER
                     0014   317
                     0014   318 BUFFER:                                  ; FAO output and other misc. buffer
             0000010E 0014   319          .BLKB   TEXT_BUFFER
                     010E   320
                     010E   321 ALT_FAO_BUF:                             ; FAO output string descriptor...
           0000 00FA 010E   322          .WORD   TEXT_BUFFER,0           ; ...during ASTs
            0000011E' 0112   323          .ADDRESS ALT_BUFFER
                     0116   324
                     0116   325 ALT_BUFFER_PTR:                          ; Fake .ASCID buffer for misc. strings
           0000 00FA 0116   326          .WORD   TEXT_BUFFER,0           ; A word for length, a word for desc.
            0000011E' 011A   327          .ADDRESS ALT_BUFFER             ; Used during ASTs
                     011E   328
                     011E   329 ALT_BUFFER:                              ; FAO output and other misc. buffer...
             00000218 011E   330          .BLKB   TEXT_BUFFER             ; ...during ASTs
                     0218   331
                     0218   332 DEVDSC:                                  ; Device name descriptor
                0000 000A   333          .WORD   MAX_DEV_DESIG,0
                 00000237' 021C   334          .ADDRESS DEV_NAME
                     0220   335
                     0220   336 PROCESS_NAME:                            ; Process name
46 50 4D 44 00000228'010E0000' 0220   337          .ASCID  /DMPF/
                     022C   338
                     0000000B 022C   339          PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
                     00000237 022C   340          .BLKB   PROCESS_NAME_FREE
                     0237   341
                     0237   342 DEV_NAME:                                ; Device name buffer
                     00000246 0237   343          .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG
                     0000000F 0246   344          NAME_LEN = .-DEV_NAME
                     0246   345
                     0246   346 DIB:                                     ; Device Information Block
                0000 0074 0246   347          .WORD   DIB$K_LENGTH,0
                 0000024E' 024A   348          .ADDRESS DIBBUF
                     024E   349 DIBBUF:
                     000002C2 024E   350          .BLKB   DIB$K_LENGTH
                     02C2   351
                     02C2   352 ERROR_COUNT:                             ; Cumulative error count at runtime
                     00000000 02C2   353          .LONG   0
                     02C6   354
                     02C6   355 STATUS:                                  ; Status value on program exit
                     00000000 02C6   356          .LONG   0
                     02CA   357
```

```
                                                     K 1
                     02CA   358 QUAD_STATUS:                        ; IO status block for misc sys. svcs.
 00000000 00000000   02CA   359         .QUAD   0
                     02D2   360
                     02D2   361 INADDRESS:                          , $CRMPSC address storage
 00000000 00000000   02D2   362         .LONG   0,0
                     02DA   363 OUTADDRESS:
 00000000 00000000   02DA   364         .LONG   0,0
                     02E2   365
                     02E2   366 UNIT_NUMBER:                        ; Current dev unit number
           0000      02E2   367         .WORD   0
                     02E4   368
                     02E4   369 DEVNAM_LEN:                         ; Current device name length
           0000      02E4   370         .WORD   0
                     02E6   371
                     02E6   372 ITERATION:                          ; # of times all tests were executed
       00000000      02E6   373         .LONG   0
                     02EA   374
                     02EA   375 PASS:                               ; Pass count
       00000000      02EA   376         .LONG   0
                     02EE   377
                     02EE   378 MSG_BLOCK:                          ; Auxiliary $GETMSG info
       000002F2      02EE   379         .BLKB   4
                     02F2   380
                     02F2   381 EXIT_DESC:                          ; Exit handler descriptor
       00000000      02F2   382         .LONG   0
       00000C26'     02F6   383         .ADDRESS EXIT_HANDLER
       00000001      02FA   384         .LONG   1
       000002C6'     02FE   385         .ADDRESS STATUS
                     0302   386
                     0302   387 ARG_COUNT:                          ; Argument counter used by ERROR_EXIT
       00000000      0302   388         .LONG   0
                     0306   389
                     0306   390 XD_CHAN:                            ; DMP/F circuit channel
           0000      0306   391         .WORD   0
                     0308   392
                     0308   393 BUF_LEN:                           ; Length of primary chars
           0000      0308   394         .WORD   0
                     030A   395
                     030A   396 BUF_DESC:                          ; Get channnel char buffer descriptor
       00000074      030A   397         .LONG   DIB$K_LENGTH
       00000312'     030E   398         .LONG   CHAN_BUF
                     0312   399
                     0312   400 CHAN_BUF:                          ; Channel char buffer
       00000386      0312   401         .BLKB   DIB$K_LENGTH
                     0386   402
                     0386   403 P1BUF:                             ; P1 Device char buffer
 00000000 00000000   0386   404         .QUAD   0
                     038E   405
                     038E   406 TR_P1BUF:                          ; p1 buufer for trib
 00000000 00000000   038E   407         .QUAD   0
                     0396   408
                     0396   409 P2BUF_DESC:                        ; P2 extended char buffer
       0000000C'     0396   410         .LONG   P2BUF_LEN
       0000039E'     039A   411         .ADDRESS P2BUF
                     039E   412
                     039E   413 P2BUF:                             ; P2 extended buffer
           0458      039E   414         .WORD   NMA$C_PCLI_PRO      ; Protocol mode
```

```
00000000  03A0   415          .LONG   NMA$C_LINPR_POI          ; DDCMP proint-to-point mode
          03A4   416
     0456 03A4   417          .WORD   NMA$C_PCLI_CON           ; Controller mode
00000001  03A6   418          .LONG   NMA$C_LINCN_LOO          ; Loopback mode
          03AA   419
0000000C  03AA   420 P2BUF_LEN = .-P2BUF
          03AA   421
          03AA   422 TR_P2BUF_DESC:                          ; P2 extended char buffer for trib
00000006' 03AA   423          .LONG   TR_P2BUF_LEN
000003B2' 03AE   424          .ADDRESS TR_P2BUF
          03B2   425
          03B2   426 TR_P2BUF:                               ; P2 extended buffer for trib
     0474 03B2   427          .WORD   NMA$C_PCCI_TRI           ; tributary address
00000001  03B4   428          .LONG   1                        ; Address
          03B8   429
00000006  03B8   430 TR_P2BUF_LEN = .-TR_P2BUF
          03B8   431
          03B8   432 SENSE_P1BUF:                            ; P1 buffer for sense mode test
00000000 00000000 03B8   433          .QUAD   0
          03C0   434
          03C0   435 SENSE_P2DESC:                           ; P2 buffer descrip for sense mode test
00000090' 03C0   436          .LONG   SENSE_P2LEN
000003C8' 03C4   437          .ADDRESS SENSE_P2BUF
          03C8   438
          03C8   439 SENSE_P2BUF:                            ; P2 buffer for sense mode test
00000458  03C8   440          .BLKW   <3*24>                   ; 8 quad quad words for dev information
00000090  0458   441 SENSE_P2LEN = .-SENSE_P2BUF             ; P2 buffer length
          0458   442
          0458   443 ERRTST_P2DESC:                          ; P2 desc for error test
00000008' 0458   444          .LONG   ERRTST_P2LEN
00000460' 045C   445          .ADDRESS ERRTST_P2BUF
          0460   446
          0460   447 ERRTST_P2BUF:                           ; P2 buffer for error test
00000468  0460   448          .BLKQ   1
00000008  0468   449 ERRTST_P2LEN = .-ERRTST_P2BUF
          0468   450
          0468   451 ERRCOUNT_DESC:                          : Error counter buffer descrip
00000200' 0468   452          .LONG   ERRCNT_LEN
00000470' 046C   453          .ADDRESS ERRCNT_BUF
          0470   454
          0470   455 ERRCNT_BUF:                             ; Buffer for error counters
00000670  0470   456          .BLKQ   64
00000200  0670   457 ERRCNT_LEN = .-ERRCNT_BUF               ; Buffer length
          0670   458
          0670   459 XD_IOSB:                                ; QIO IO status block for transmit
00000678  0670   460          .BLKQ   1
          0678   461
          0678   462 RCV_IOSB:                               ; QIO Io status block for receive
00000680  0678   463          .BLKQ   1
          0680   464
          0680   465 XMIT_BUF:                               ; Transmit buffer
00000880  0680   466          .BLKB   MAX_MSG_LEN
          0880   467
          0880   468 RECV_BUF:                               ; Receive buffer
00000A80  0880   469          .BLKB   MAX_MSG_LEN
          0A80   470
          0A80   471 BAD_DATA:                               ; Received wrong data
```

```
                     00  0A80   472          .BYTE   0
                         0A81   473
                         0A81   474 GOOD_DATA:                                ; Data sent (good)
                     00  0A81   475          .BYTE   0
                         0A82   476
                         0A82   477
                         0A82   478 ;
                         0A82   479 ; Head of self-relative UETP unit block queue.
                         0A82   480 ;
                         0A82   481          .ALIGN QUAD
                         0A88   482
                         0A88   483 UNIT_LIST:                                ; Head of unit block circular list
          00000000 00000000  0A88   484          .QUAD   0
                         0A90   485
                         0A90   486 NEW_NODE:                                 ; Newly acquired node address
          00000000 00000000  0A90   487          .QUAD   0
```

```
                                     N 1
                     0A98   489          .SBTTL  RMS-32 Data Structures
                     0A98   490          .ALIGN  LONG
                     0A98   491
                     0A98   492  SYSIN_FAB:                                ; Allocate FAB for SYS$INPUT
                     0A98   493          $FAB-
                     0A98   494          FNM = <SYS$INPUT>
                     0AE8   495
                     0AE8   496  SYSIN_RAB:                                ; Allocate RAB for SYS$INPUT
                     0AE8   497          $RAB-
                     0AE8   498          FAB = SYSIN_FAB,-
                     0AE8   499          ROP = PMT,-
                     0AE8   500          PBF = PROMPT,-
                     0AE8   501          PSZ = PMTSIZ,-
                     0AE8   502          UBF = DEV_NAME,-
                     0AE8   503          USZ = NAME_LEN
                     0B2C   504
                     0B2C   505  INI_FAB:                                ; Allocate FAB for UETINIDEV
                     0B2C   506          $FAB-
                     0B2C   507          FAC = <GET,PUT,UPD>,-
                     0B2C   508          RAT = CR,-
                     0B2C   509          SHR = <GET,PUT,UPI>,-
                     0B2C   510          FNM = <UETINIDEV.DAT>
                     0B7C   511
                     0B7C   512  INI_RAB:                                ; Allocate RAB for UETINIDEV
                     0B7C   513          $RAB-
                     0B7C   514          FAB = INI_FAB,-
                     0B7C   515          RBF = BUFFER,-
                     0B7C   516          UBF = BUFFER,-
                     0B7C   517          USZ = REC_SIZE
                     0BC0   518
                     0BC0   519  DDB_RFA:                                 ; RFA storage for INI_RAB
00000BC6             0BC0   520          .BLKB   6
                     0BC6   521
                     0BC6   522          .ALIGN  LONG
                     0BC8   523  SUP_FAB:                                ; Allocate FAB for UETSUPDEV
                     0BC8   524          $FAB-
                     0BC8   525          FAC = GET,-
                     0BC8   526          SHR = <UPI,GET>,-
                     0BC8   527          RAT = CR,-
                     0BC8   528          FOP = UFO,-
                     0BC8   529          FNM = <UETSUPDEV.DAT>
                     0C18   530
                     0C18   531  ;
                     0C18   532  ; Dummy FAB and RAB to copy to the UETP unit blocks
                     0C18   533  ; The following FAB and RAB must be contiguous and in this order!
                     0C18   534  ;
                     0C18   535
                     0C18   536  DUMMY_FAB:
                     0C18   537          $FAB
                     0C68   536
                     0C68   539  DUMMY_RAB:
                     0C68   540          $RAB    RSZ = WRITE_SIZE,-
                     0C68   541                  USZ = READ_SIZE
```

```
                     0CAC   543          .SBTTL  Main Program
                 00000000   544          .PSECT  DMPF,EXE,NOWRT,PAGE
                     0000   545
                     0000   546          .DEFAULT DISPLACEMENT,WORD
                     0000   547
                0000 0000   548 .ENTRY UETDMPF00,^M<>                       ; Entry mask
                     0002   549
    6D  09D1'CF      DE 0002   550          MOVAL   SSERROR,(FP)            ; Declare exception handler
                        0007   551          $SETSFM_S ENBFLG = #1           ; Enable system service failure mode
                        0010   552          $DCLEXH_S DESBLK = EXIT_DESC    ; Declare an exit handler
                        001B   553
                        001B   554          $OPEN   FAB = SYSIN_FAB,-       ; Open SYS$INPUT
                        001B   555                  ERR = RMS_ERROR
                        002A   556          $CONNECT RAB = SYSIN_RAB,-      ; Connect RAB to SYS$INPUT
                        002A   557                   ERR = RMS_ERROR
          02    E1 0039   558          BBC     S^#DEV$V_TRM,-          ; BR if SYS$INPUT is NOT a terminal
    1E 0AD8'CF       003B   559                  SYSIN_FAB+FAB$L_DEV,10$
                        003F   560          $TRNLOG_S LOGNAM = CONTROLLER,- ; Allow terminal user to specify...
                        003F   561                    RSLLEN = DEVNAM_LEN,- ; ...a logial name...
                        003F   562                    RSLBUF = DEVDSC       ; ...for the controller to test
       01    50    D1 0058   563          CMPL    R0,#SS$_NORMAL          ; Was a controller specified?
             2E    13 005B   564          BEQL    PROC_CONT_NAME          ; BR if it was - go process it
                     005D   565 10$:
                        005D   566          $GET    RAB = SYSIN_RAB,-       ; Read SYS$INPUT...
                        005D   567                  ERR = RMS_ERROR         ; ...for the controller name
       0B0A'CF    B0 006C   568          MOVW    SYSIN_RAB+RAB$W_RSZ,-   ; Save the name length
       02E4'CF       0070   569                  DEVNAM_LEN
             16    12 0073   570          BNEQ    PROC_CONT_NAME          ; BR if we got something
    02C6'CF    14    D0 0075   571          MOVL    #SS$_BADPARAM,STATUS    ; Save an exit status if not
       00C4'CF    DF 007A   572          PUSHAL  NO_CTRLNAME             ; Prepare for message...
             01    DD 007E   573          PUSHL   #1                      ; ...arg count
     00741132 8F    DD 0080   574          PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...signal name
             03    DD 0086   575          PUSHL   #3                      ; ...arg count
              0AF6   31 0088   576          BRW     ERROR_EXIT              ; ...go tell of bad setup
                     008B   577
                     008B   578 PROC_CONT_NAME:
    0218'CF    02E4'CF    3C 008B   579          MOVZWL  DEVNAM_LEN,DEVDSC       ; Set the device name length
          0218'CF    DF 0092   580          PUSHAL  DEVDSC                  ; Make sure...
          0218'CF    DF 0096   581          PUSHAL  DEVDSC                  ; ...that the specified controller...
     00000000'GF    02 009A   582          CALLS   #2,G^STR$UPCASE         ; ...is all uppercase for later comaparison
52   0218'CF    01    C1 00A1   583          ADDL3   #1,DEVDSC,R2            ; Estimate the eventual...
       022C'CF    52    A0 00A7   584          ADDW2   R2,PROCESS_NAME         ; ...process name length (incl. "_")
                     DE 00AC   585          MOVAL   PROCESS_NAME+8-         ; Locate first available byte...
                     00AD   586                  +MAX_PROC_NAME-         ; ...in process name handle...
       50    022C'CF       00AD   587                  -PROCESS_NAME_FREE,R0   ; ...for device name
             0B    C3 00B1   588          SUBL3   #PROCESS_NAME_FREE,-    ; Will the device name fit...
          51    52       00B3   589                  R2,R1                   ; ...in the remaining space?
             08    15 00B5   590          BLEQ    10$                     ; BR if it will
          50    51    C2 00B7   591          SUBL2   R1,R0                       ; Overwrite handle otherwise...
       022C'CF    0F    B0 00BA   592          MOVW    #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
                     00BF   593 10$:
         80    5F 8F    90 00BF   594          MOVB    #^A/_/,(R0)+            ; Separate handle from device name
60   0237'CF    0218'CF    28 00C3   595          MOVC3   DEVDSC,DEV_NAME,(R0)    ; Concatenate handle with device name
                7E    D4 00CB   596          CLRL    -(SP)                   ; Set the time stamp flag
          000F'CF    DF 00CD   597          PUSHAL  TEST_NAME               ; Set the test name
                02    DD 00D1   598          PUSHL   #2                      ; Push the argument count
     00741039 8F    DD 00D3   599          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
```

```
00000000'GF   04   FB  00D9   600          CALLS   #4,G^LIB$SIGNAL          ; Print the startup message
    0002'CF   08   A8  00E0   601          BISW2   #BEGIN_MSGM,FLAG         ; Set flag so we don't print it again
                       00E5   602          $SETPRN_S PRCNAM = PROCESS_NAME ; Set the process name to UETDMPF00_x
                       00F0   603
                  02   E1  00F0   604          BBC     S^#DEV$V_TRM,-           ; BR if SYS$INPUT is NOT a terminal
         66 0AD8'CF        00F2   605                  SYSIN_FAB+FAB$L_DEV,20$
                       00F6   606          $GETDVI_S DEVNAM = SYS$INPUT,-   ; Get the name of...
                       00F6   607                    EFN    = #SS_SYNCH_EFN,- ; ...device which may abort test
                       00F6   608                    ITMLST = INPUT_ITMLST,-
                       00F6   609                    IOSB   = QUAD_STATUS
         45 02CA'CF   E9  0112   610          BLBC    QUAD_STATUS,20$          ; Avoid CTRL/C handler if any error
                       0117   611          $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handler
                       0117   612                    CHAN   = TTCHAN
                       0128   613          $QIOW_S CHAN    = TTCHAN,-       ; Enable CTRL/C AST's...
                       0128   614                  FUNC    = #IO$_SETMODE!IO$M_CTRLCAST,-
                       0128   615                  P1      = CCASTHAND
            0220'CF   DF  0149   616          PUSHAL  PROCESS_NAME             ; ...and tell the user...
                  01   DD  014D   617          PUSHL   #1                       ; ...
       0074832B 8F   DD  014F   618          PUSHL   #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
00000000'GF   03   FB  0155   619          CALLS   #3,G^LIB$SIGNAL          ; ...
                       015C   620 20$:
```

```
                     015C   622 ;
                     015C   623 ; From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
                     015C   624 ; and unit configuration and lets us know if the setup to run this test was
                     015C   625 ; done correctly.
                     015C   626 ;
                     015C   627         $OPEN   FAB = INI_FAB,-          ; Open file 'UETINIDEV.DAT"
                     015C   628                 ERR = RMS_ERROR
                     016B   629         $CONNECT RAB = INI_RAB,-         ; Connect the RAB and FAB
                     016B   630                  ERR = RMS_ERROR
                     017A   631         $MGBLSC_S  INADR = INADDRESS,-   ; Connect to UETSUPDEV global section
                     017A   632                    RETADR = OUTADDRESS,-
                     017A   633                    GSDNAM = SUPDEV_GBLSEC,-
                     017A   634                     FLAGS = #SEC$M_EXPREG
          00000978 8F 50 D1  0199   635         CMPL    R0,#SS$_NOSUCHSEC        ; Was the section already there?
                   37 12    01A0   636         BNEQ    30$                      ; BR if it was...
                     01A2   637         $OPEN   FAB = SUP_FAB,-          ; ...else open "UETSUPDEV.DAT"
                     01A?   638                 ERR = RMS_ERROR
                     01B1   639         $CRMPSC_S CHAN = SUP_FAB+FAB$L_STV,- ; Create the global section
                     01B1   640                   INADR = INADDRESS,-
                     01B1   641                  RETADR = OUTADDRESS,-
                     01B1   642                  GSDNAM = SUPDEV_GBLSEC,-
                     01B1   643                   FLAGS = #SEC$M_EXPREG!SEC$M_GBL
                     01D9   644 30$:
     56 02DE'CF 02DA'CF C3  01D9   645         SUBL3   OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length
                     01E1   646
                     01E1   647 FIND_IT:
                     01E1   648         $GET    RAB = INI_RAB,-          ; Get the first record
                     01E1   649                 ERR = RMS_ERROR
          01F5'CF    DF  01F0   650         PUSHAL  CONT_DESC                ; Make sure...
          01F5'CF    DF  01F4   651         PUSHAL  CONT_DESC                ; ...that the controller name...
    00000000'GF    02  FB  01F8   652         CALLS   #2,G^STR$UPCASE          ; ...is all uppercase letters
     0014'CF    44 8F  91  01FF   653         CMPB    #^A/D/,BUFFER            ; Is this a DDB?
                   27  13  0205   654         BEQL    10$                      ; Go on if not
     0014'CF    45 8F  91  0207   655         CMPB    #^A/E/,BUFFER            ; Is this the end of the file?
                   D2  12  020D   656         BNEQ    FIND_IT                  ; Continue on if not
          0218'CF    DF  020F   657         PUSHAL  DEVDSC                   ; Push device not supported message
          0220'CF    DF  0213   658         PUSHAL  PROCESS_NAME             ; Parameters on the stack
                   02  DD  0217   659         PUSHL   #2
       00748333 8F  DD  0219   660         PUSHL   #UETP$_DENOSU
                   02  F0  021F   661         INSV    #STS$K_ERROR,-           ; Set the severity code...
                   00      0221   662                 #STS$V_SEVERITY,-
                   6E  03  0222   663                 #STS$S_SEVERITY,(SP)
          02C6'CF    6E  D0  0224   664         MOVL    (SP),STATUS              ; ...and save it as the exit status
                   04  DD  0229   665         PUSHL   #4
                 0953  31  022B   666         BRW     ERROR_EXIT               ; Exit in error
                     022E   667 10$:
0237'CF 001A'CF 02E4'CF 29  022E   668         CMPC    DEVNAM_LEN,BUFFER+6,DEV_NAME     ; Is this the right controller?
                   A7  12  0238   669         BNEQ    FIND_IT                  ; BR if not
     0BC0'CF 0B8C'CF   06  28  023A   670         MOVC3   #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the Record File Address
          0018'CF    54 8F  91  0242   671         CMPB    #^A/T/,BUFFER+4          ; Can we test this controller?
                   2F  13  0248   672         BEQL    FOUND_IT                 ; BR if we can...
                     024A   673         $FAO_S  CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if we can't
                     024A   674                 OUTLEN = BUFFER_PTR,-
                     024A   675                 OUTBUF = FAO_BUF,-
                     024A   676                 P1     = #DEVDSC
          02C6'CF    14  D0  0263   677         MOVL    #SS$_BADPARAM,STATUS     ; Set return status
               000C'CF  DF  0268   678         PUSHAL  BUFFER_PTR               ; ...
```

```
                01   DD  026C   679          PUSHL   #1                              ; ...
        00741132 8F   DD  026E   680          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; ...
                03   DD  0274   681          PUSHL   #3                              ; ...
              0908   31  0276   682          BRW     ERROR_EXIT                      ; We can't test what we can't test
                         0279   683
                         0279   684 FOUND_IT:
                         0279   685          $GET    RAB = INI_RAB,-                 ; Get a record
                         0279   686                  ERR = RMS_ERROR
            01F5'CF   DF  0288   687          PUSHAL  CONT_DESC                       ; Make sure...
            01F5'CF   DF  028C   688          PUSHAL  CONT_DESC                       ; ...that this line...
00000000'GF   02   FB  0290   689          CALLS   #2,G^STR$UPCASE                 ; ...is all uppercase letters
 0014'CF   55 8F   91  0297   690          CMPB    #^A/U/,BUFFER                   ; Is this a UCB?
                24   13  029D   691          BEQL    30$                             ; BR if it is
 0014'CF   44 8F   91  029F   692          CMPB    #^A/D/,BUFFER                   ; Is this a DDB?
                19   13  02A5   693          BEQL    20$                             ; BR if yes
 0014'CF   45 8F   91  02A7   694          CMPB    #^A/E/,BUFFER                   ; Is this the end?
                11   13  02AD   695          BEQL    20$                             ; BR if yes
                         02AF   696 10$:
            0151'CF   DF  02AF   697          PUSHAL  ILLEGAL_REC                     ; Then this is an error in the record
                01   DD  02B3   698          PUSHL   #1                              ; Push the error message
        00741132 8F   DD  02B5   699          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; Push the signal name
                03   DD  02BB   700          PUSHL   #3                              ; Push the temp arg count
              08C1   31  02BD   701          BRW     ERROR_EXIT                      ; Finish for good
                         02C0   702 20$:
              0123   31  02C0   703          BRW     ALL_SET                         ; Found DDB or END
                         02C3   704 30$:
 0018'CF   54 8F   91  02C3   705          CMPB    #^A/T/,BUFFER+4                 ; Is the unit testable?
                AE   12  02C9   706          BNEQ    FOUND_IT                        ; BR if not
                01   DD  02CB   707          PUSHL   #1                              ; Flag to ignore blanks when converting
                02   DD  02CD   708          PUSHL   #2                              ; Set byte size of results
            02E2'CF   DF  02CF   709          PUSHAL  UNIT_NUMBER                     ; Set address to receive word
            01ED'CF   DF  02D3   710          PUSHAL  UNIT_DESC                       ; Push string address
      00000000'GF   04   FB  02D7   711          CALLS   #4,G^OTS$CVT_TI_L               ; Convert ASCII unit # to decimal
                CE   50   E9  02DE   712          BLBC    R0,10$                          ; Don't allow bogus unit to pass
                05   20   3B  02E1   713          SKPC    #^A/ /,#MAX_UNIT_DESIG,- ; Find out where unit number really is
            001A'CF           02E4   714                  BUFFER+6
                50   D7  02E7   715          DECL    R0                              ; Units must all be at least one digit
          61    50   30  02E9   716          SKPC    #^A/0/,R0,(R1)                  ; Skip leading zeroes on the unit
                50   D6  02ED   717          INCL    R0                              ; Compensate for DECL above
  0218'CF   02E4'CF   50   A1  02EF   718          ADDW3   R0,DEVNAM_LEN,DEVDSC            ; Calculate device'unit string length
          52    02E4'CF   3C  02F7   719          MOVZWL  DEVNAM_LEN,R2                   ; Offset to unit number in DEVDSC
       0237'C2  61   50   28  02FC   720          MOVC3   R0,(R1),DEV_NAME(R2)            ; Append unit number to device
                         0302   721          $GETDEV_S DEVNAM = DEVDSC,-             ; Get the device characteristics
                         0302   722                    PRIBUF = DIB
          57    0252'CF   9A  0317   723          MOVZBL  DIBBUF+DIB$B_DEVCLASS,R7 ; Save the device class
          58    0253'CF   9A  031C   724          MOVZBL  DIBBUF+DIB$B_DEVTYPE,R8 ; Save the device type
                         0321   725          $FAO_S  CTRSTR = CS1,-
                         0321   726                  OUTBUF = FAO_BUF,-
                         0321   727                  P1     = R7,-
                         0321   728                  P2     = R8                     ; Make it into a string
02DA'DF   56   0014'CF   06   39  0336   729          MATCHC  #6,BUFFER,R6,@OUTADDRESS ; Find the device class and type
                1E   13  033F   730          BEQL    40$                             ; BR if it was found
                         0341   731          $FAO_S  CTRSTR = CS3,-                  ; Try for full class support
                         0341   732                  OUTBUF = FAO_BUF,-
                         0341   733                     P1 = R7
02DA'DF   56   0014'CF   06   39  0354   734          MATCHC  #6,BUFFER,R6,@OUTADDRESS ; Find the device class only
                0D   12  035D   735          BNEQ    50$                             ; BR if not found
```

```
                     035F    736 40$:
      55  000F'CF    9A 035F    737          MOVZBL  TEST_NAME,R5             ; Get the test name length
0017'CF   63    55   29 0364    738          CMPC3   R5,(R3),TEST_NAME+8     ; Are we the right test?
                1F   13 036A    739          BEQL    60$                     ; BR if yes
                     036C    740 50$:
          0218'CF   DF 036C    741          PUSHAL  DEVDSC                  ; Push device not supported message
          0220'CF   DF 0370    742          PUSHAL  PROCESS_NAME            ; Parameters on the stack
                02  DD 0374    743          PUSHL   #2                      ; Push the argument count
    00748333  8F    DD 0376    744          PUSHL   #UETP$_DENOSU
                02  F0 037C    745          INSV    #STS$K_ERROR,-
                00         037E    746                  #STS$V_SEVERITY,-
                6E  03         037F    747                  #STS$S_SEVERITY,(SP)    ; Set the severity code...
        02C6'CF    6E   D0 0381    748          MOVL    (SP),STATUS             ; ...and save it as the exit status
                04  DD 0386    749          PUSHL   #4                      ; Push the partial arg count...
              07F6  31 0388    750          BRW     ERROR_EXIT              ; ...and split this scene
```

```
038B   752 ;+
038B   753 ; The following code dynamically allocates enough memory for a unit block,
038B   754 ; a device dependent parameter area and I/O buffers. The unit block is inserted
038B   755 ; into the queue header UNIT_LIST.  It then initializes the unit block.
038B   756 ; A comment indicates where the device dependent parameters should be
038B   757 ; initialized.  The unit block format is as follows:
038B   758 ;
038B   759 ;                          +----------------+ -------
038B   760 ;       UETUNT$L_FLINK     |                |       ^
038B   761 ;                          +----------------+       |
038B   762 ;       UETUNT$L_BLINK     |                |       |
038B   763 ;                          +-------+--------+---+   |
038B   764 ;       UETUNT$B_TYPE              |        |   |   |
038B   765 ;                          +-------+--------+---+   |
038B   766 ;       UETUNT$W_SIZE      |       |        |       |  contains DEVDEP_SIZE + UETUNT$C_INDSIZ
038B   767 ;                          +---+---+--------+       |
038B   768 ;       UETUNT$B_FLAGS     |   |                    |
038B   769 ;                          +---+   +--------+---+   |
038B   770 ;       UETUNT$W_CHAN              |        |   |   |
038B   771 ;                          +-------+--------+---+   |
038B   772 ;       UETUNT$W_FUNC      |       |        |       |
038B   773 ;                          +-------+--------+       +----- UETUNT$C_SIZE
038B   774 ;       UETUNT$L_ITER      |                |       |
038B   775 ;                          +----------------+       |
038B   776 ;       UETUNT$T_FILSPC    |                |       |
038B   777 ;                          |/\/\/\/\/\/\/\/\|       |
038B   778 ;                          NAM$C_MAXRSS bytes       |
038B   779 ;                          |/\/\/\/\/\/\/\/\|       |
038B   780 ;                          |                |       |
038B   781 ;       UETUNT$K_FAB       |----------------|       |
038B   782 ;                          |                |       |
038B   783 ;                          |/\/\/\/\/\/\/\/\|       |
038B   784 ;                           FAB$C_BLN bytes         |
038B   785 ;                          |/\/\/\/\/\/\/\/\|       |
038B   786 ;                          |                |       |
038B   787 ;          UETUNT$K_RAB    +----------------+       |
038B   788 ;                          |                |       |
038B   789 ;                          |\/\/\/\/\/\/\/\/|       |
038B   790 ;                           RAB$C_BLN bytes         |
038B   791 ;                          |\/\/\/\/\/\/\/\/|       |
038B   792 ;                          |                |       v
038B   793 ;       UETUNT$K_DEVDEP    +----------------+ -------
038B   794 ;                          |                |       ^
038B   795 ;                          |\/\/\/\/\/\/\/\/|       |
038B   796 ;                            user defined           +----- DEVDEP_SIZE
038B   797 ;                          |\/\/\/\/\/\/\/\/|       |
038B   798 ;                          |                |       v
038B   799 ;                          +----------------+ -------
038B   800 ;    READ/WRITE buffers     |                |       ^
038B   801 ;                          |\/\/\/\/\/\/\/\/|       |
038B   802 ;                            user defined           +----- WRITE_SIZE and READ_SIZE
038B   803 ;                          |/\/\/\/\/\/\/\/\|       |
038B   804 ;                          |                |       v
038B   805 ;                          +----------------+ -------
038B   806 ;-
```

```
                                038B    808 60$:
                                038B    809             $EXPREG_S PAGCNT = #PAGES,-     ; Get a new node of demand zero memory
                                038B    810                       RETADR = NEW_NODE
0A88'CF    0A90'DF    5D    039C    811             INSQTI  @NEW_NODE,UNIT_LIST     ; Put the new node in the unit list
     56    0A90'CF    D0    03A3    812             MOVL    NEW_NODE,R6             ; Save a copy of its address
       08 A6    01    90    03A8    813             MOVB    #1,UETUNT$B_TYPE(R6)    ; Set the structure type
          01A4 8F    B0    03AC    814             MOVW    #UETUNT$C_IRDSIZ+DEVDEP_SIZE,-
            09 A6          03B0    815                     UETUNT$W_SIZE(R6)       ; Set the structure size
  14 A6    0218'CF    90    03B2    816             MOVB    DEVDSC,UETUNT$T_FILSPC(R6) ; Set the device name size
021C'DF    0218'CF    28    03B8    817             MOVC3   DEVDSC,@DEVDSC+4,-
            15 A6          03BF    818                     UETUNT$T_FILSPC+1(R6)   ; Save the device name
          0094 8F    28    03C1    819             MOVC3   #FAB$C_BLN+RAB$C_BLN,-
0110 C6    0C18'CF        03C5    820                     DUMMY_FAB,UETUNT$C_FAB(R6) ; Save a FAB and a RAB away
     57    0110 C6    DE    03CB    821             MOVAL   UETUNT$K_FAB(R6),R7     ; Save the FAB address
     58    0160 C6    DE    03D0    822             MOVAL   UETUNT$K_RAB(R6),R8     ; Save the RAB address
       3C A8    57    D0    03D5    823             MOVL    R7,RAB$L_FAB(R8)        ; Set the FAB address in the RAB
            14 A6    90    03D9    824             MOVB    UETUNT$T_FILSPC(R6),-
            34 A7          03DC    825                     FAB$B_FNS(R7)           ; Set the FNS field in the FAB
            15 A6    DE    03DE    826             MOVAL   UETUNT$T_FILSPC+1(R6),-
            2C A7          03E1    827                     FAB$L_FNA(R7)           ; Set the FNA field in the FAB
                           03E3    828 ;
                           03E3    829 ; Set the device dependent parameters in here
                           03E3    830 ;
              FE93    31   03E3    831             BRW     FOUND_IT                ; Do the next UCB
```

```
                     03E6    833 ;
                     03E6    834 ; Arrive here when we have the device configuration.  In normal or loop forever
                     03E6    835 ; mode, set a timer far enough in the future such that we can do a reasonable
                     03E6    836 ; set of tests before the timer expires, but if our device gets hung, the
                     03E6    837 ; program won't waste too much time before noticing.  Let one-shot mode be a
                     03E6    838 ; special case.
                     03E6    839 ;
                     03E6    840 ALL_SET:
          0A88'CF    D5  03E6    841          TSTL    UNIT_LIST                ; Anything to test?
                16   12  03EA    842          BNEQ    10$                      ; BR if yes
          012B'CF    DF  03EC    843          PUSHAL  NOUNIT_SELECTED          ; Else set up the error message...
                01   DD  03F0    844          PUSHL   #1                       ; ...argument count...
           00741132  8F  DD  03F2    845          PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...signal name...
                03   DD  03F8    846          PUSHL   #3                       ; ...and parameter count
          02C6'CF    14  D0  03FA    847          MOVL    #SS$_BADPARAM,STATUS     ; Set return status
                077F   31  03FF    848          BRW     ERROR_EXIT               ; ...and give up, complaining
                     0402    849 10$:
          0002'CF    04  A8  0402    850          BISW2   #SAFE_TO_UPDM,FLAG       ; OK safe to update UETINIDEV.DAT now
```

```
                     0407   852          .SBTTL  Test the DMP/DMF
                     0407   853 ;
                     0407   854 START_TEST:
                     0407   855
                     0407   856          $ASSIGN_S -                         ; Assign channel to the device
                     0407   857                  DEVNAM = DEVDSC,-
                     0407   858                  CHAN = XD_CHAN
                     0418   859
                     0418   860
          22 50   E8 0418   861          BLBS    R0,10$                      ; BR if no failure
02C6'CF   50   D0 041B   862          MOVL    R0,STATUS                   ; Save the failure status
     02C6'CF   DD 042C   863          PUSHL   STATUS                      ; Push the error code...
     02C6'CF   DD 0424   864          PUSHL   STATUS
     0218'CF   DF 0428   865          PUSHAL  DEVDSC                      ; ...and the device designation...
     000F'CF   DF 042C   866          PUSHAL  TEST_NAME                   ; ...and the test name...
           03  DD 0430   867          PUSHL   #3                          ; ...and the arg count...
 0074819A 8F   DD 0432   868          PUSHL   #UETP$_DEUNUS!STS$K_ERROR   ; ...and the signal name...
           06  DD 0438   869          PUSHL   #6                          ; ...and the total argument count...
         0744  31 043A   870          BRW     ERROR_EXIT                  ; ...and bail out completely
                     043D   871 10$:
                     043D   872 ;
                     043D   873 ; Set up P1 device char buffer, P2 buffer is set up in Read/write section
                     043D   874 ;
                     043D   875 RESTART:
53   0388'CF   DE 043D   876          MOVAL   P1BUF+2,R3                  ; Address of device char for p1
83   0200 8F   B0 0442   877          MOVW    #MAX_MSG_LEN,(R3)+          ; Maximum message length
     63   02   90 0447   878          MOVB    #XM$M_CHR_LOOPB,(R3)        ; Set loop back mode in char
                     044A   879
                     044A   880          $SETIMR_S -                         ; Set up half minute timer
                     044A   881                  DAYTIM = HALFMIN,-          ; to prevent hung
                     044A   882                  ASTADR = TIME_ERR_OUT,-
                     044A   883                  REQIDT = #START_TO_MSG
                     0461   884 START_CONT:
                     0461   885          $QIOW_S -                           ; Start the controller
                     0461   886                  CHAN = XD_CHAN,-
                     0461   887                  FUNC = #IO$_SETMODE!IO$M_CTRL!IO$M_STARTUP,-
                     0461   888                  IOSB = XD_IOSB,-
                     0461   889                  ASTADR = CHK_QIO_AST,-
                     0461   890                  ASTPRM = #START_CONT_PRM,-
                     0461   891                  P1 = P1BUF,-
                     0461   892                  P2 = #P2BUF_DESC,-
                     0461   893                  P3 = #RECVPOOL_SIZ
                     0492   894
                     0492   895 START_TRI:
                     0492   896          $QIOW_S -                           ; Start the tributary
                     0492   897                  CHAN = XD_CHAN,-
                     0492   898                  FUNC = #IO$_SETMODE!IO$M_STARTUP,-
                     0492   899                  IOSB = XD_IOSB,-
                     0492   900                  ASTADR = CHK_QIO_AST,-
                     0492   901                  ASTPRM = #START_TRIB_PRM,-
                     0492   902                  P1 = TR_P1BUF,-
                     0492   903                  P2 = #TR_P2BUF_DESC,-
                     0492   904                  P3 = #RECVPOOL_SIZ          ; Common receive pool = 4 buffer
                     04C3   905
0002'CF   20   A8 04C3   906          BISW2   #FLAG_SHUTDNM,FLAG          ; Set flag to say shut down the
                     04C8   907                                              ; device if errors occur
                     04C8   908
```

```
                         04C8           $CANTIM_S REQIDT = #START_TO_MSG ; Cancel hung timer
                         04D7
                         04D7           $TRNLOG_S LOGNAM = MODE,-       ; Get the run mode
                         04D7                   RSLLEN = BUFFER_PTR,-
                         04D7                   RSLBUF = FAO_BUF
                         04F0
    0014'CF    20    8A  04F0           BICB2   #LC_BITM,BUFFER         ; Convert to upper case
  0014'CF   4F 8F    91  04F5           CMPB    #^A70/,BUFFER           ; Is this a one shot?
               0D    12  04FB           BNEQ    10$                     ; BR if not
    0002'CF    02    A8  04FD           BISW2   #TEST_OVERM,FLAG        ; End after one iteration
    0002'CF    10    A8  0502           BISW2   #MODE_IS_ONEM,FLAG      ; Set mode is "ONE" flag
             0013    31  0507           BRW     LOOPBACK_TEST           ; Skip the 3 min timer, mode is "one"
                         050A  10$:                                     ; Not one shot
                         050A           $SETIMR_S DAYTIM = THREEMIN,-   ; Set 3 minutes timer for xmit/recv
                         050A                   ASTADR = TIME_SUC_OUT ; The test will do xmit/recv for about
                         051D                                           ; 3 minutes
                         051D  ;
                         051D  ; Loopback test transmit and receive random data with different message length
                         051D  ;
                         051D  LOOPBACK_TEST:
     52   AA 8F      9A  051D           MOVZBL  #^XAA,R2                ; Random number 1
        53   2E      9A  0521           MOVZBL  #^X2E,R3                ; Random number 2
 57   00000200 8F    D0  0524           MOVL    #MAX_MSG_LEN,R7         ; Maximum message length
                         052B  SET_XMIT_BUF:                            ; Set up transmit buffer
     56   0680'CF    DE  052B           MOVAL   XMIT_BUF,R6             ; Transmit buffer address
        54   57      D0  0530           MOVL    R7,R4                   ; Message length in bytes
                         0533  10$:
        52   53      C0  0533           ADDL2   R3,R2                   ; Random number as data
        86   52      90  0536           MOVB    R2,(R6)+                ; Fill in the transmit buffer
          F7 54      F5  0539           SOBGTR  R4,10$                  ; Branch if more bytes to be filled
                         053C
                         053C           $SETIMR_S -                     ; Set half minute timer to prevent hung
                         053C                   DAYTIM = HALFMIN,-
                         053C                   ASTADR = TIME_ERR_OUT,-
                         053C                   REQIDT = #RW_TO_MSG
                         0553
        58   10      D0  0553           MOVL    #LIMIT,R8               ; Loop 16 times for each msg length
                         0556  XMIT:
                         0556           $QIO_S -                        ; Transmit data message
                         0556                   EFN = #XMIT_EFN,-       ; Event flag
                         0556                   CHAN = XD_CHAN,-        ; Channel
                         0556                   FUNC = #IO$_WRITEVBLK,- ; Transmit
                         0556                   IOSB = XD_IOSB,-        ; IOSB
                         0556                   ASTADR = CHK_QIO_AST,-  ; Completion ast routine
                         0556                   ASTPRM = #WRITE_PRM,-   ; Ast parameter
                         0556                   P1 = XMIT_BUF,-         ; Addr of transmit buffer
                         0556                   P2 = R7                 ; message length in bytes
                         0581
                         0581  RECV:
                         0581           $QIOW_S -                       ; Read data message
                         0581                   EFN = #RECV_EFN,-       ; Event flag
                         0581                   CHAN = XD_CHAN,-        ; Channel
                         0581                   FUNC = #IO$_READVBLK,-  ; Receive message
                         0581                   IOSB = RCV_IOSB,-       ; IOSB
                         0581                   ASTADR = RECV_AST,-     ; Completion ast to check data received
                         0581                   ASTPRM = R7,-           ; Ast parameter = message length
                         0581                   P1 = RECV_BUF,-         ; Receive buffer
```

```
                     0581   966                 P2 = R7                         ; Message length in bytes
                     05A8   967
          02E6'CF    D6  05A8   968         INCL    ITERATION                  ; Increment iteration count
                     05AC   969
                     05AC   970         $WAITFR_S EFN = #XMIT_EFN              ; Wait until transmit done
                     05B5   971
            9E 58    F5  05B5   972         SOBGTR  R8,XMIT                    ; Loop for 16 times
                     05B8   973
                     05B8   974         $CANTIM_S -                            ; Cancel hung timer
                     05B8   975                 REQIDT = #RW_TO_MSG
                     05C7   976
      09 0002'CF    01   E0  05C7   977         BBS     #TEST_OVERV,FLAG,SENSE_TEST ; Is the test over?
              03 57    F5  05CD   978         SOBGTR  R7,10$                     ; Decrement message length by one and
                     05D0   979                                                ; try again
               FF4A    31  05D0   980         BRW     LOOPBACK_TEST              ; Re-try from beginning
               FF55    31  05D3   981 10$:    BRW     SET_XMIT_BUF               ; Set  new data in tranmit buffer
                     05D6   982
                     05D6   983 SENSE_TEST:
                     05D6   984         $QIOW_S -                              ; Read device (trib.) charracteristic
                     05D6   985                 CHAN = XD_CHAN,-
                     05D6   986                 FUNC = #IO$_SENSEMODE,-
                     05D6   987                 IOSB = XD_IOSB,-
                     05D6   988                 P1 = SENSE_P1BUF,-
                     05D6   989                 P2 = #SENSE_P2DESC
                     05FD   990
                     05FD   991         $FAO_S  CTRSTR = SENSE_PRM,-
                     05FD   992                 OUTLEN = ALT_BUFFER_PTR,-
                     05FD   993                 OUTBUF = ALT_FAO_BUF,-
                     05FD   994                 P1     = #DEVDSC
          0116'CF    DF  0616   995         PUSHAL  ALT_BUFFER_PTR
      08A5'CF    01   FB  061A   996         CALLS   #1,CHECK_IOSB              ; Check status
                     061F   997
      54  0672'CF    3C  061F   998         MOVZWL  XD_IOSB+2,R4               ; Number of bytes returned for p2 buff
      55  03B2'CF    DE  0624   999         MOVAL   TR_P2BUF,R5                ; Address of P2 buff
          57   06    D0  0629  1000         MOVL    #TR_P2BUF_LEN,R7           ; P2 length
                     062C  1001 10$:
      56  03C8'CF    DE  062C  1002         MOVAL   SENSE_P2BUF,R6             ; Address of P2 buff returned
66    54  65   06    39  0631  1003         MATCHC  #6,(R5),R4,(R6)            ; Check the parameters returned
               0C    12  0636  1004         BNEQ    30$                        ; Br if not match
        55   06 A5    DE  0638  1005         MOVAL   6(R5),R5                   ; Next parameter
          57   06    C2  063C  1006         SUBL2   #6,R7                      ; Index
               EB    12  063F  1007         BNEQ    10$                        ; Br if more parameters to check
             001F    31  0641  1008         BRW     ERROR_TEST                 ; Otherwise go to test error case
                     0644  1009 30$:
                     0644  1010         $FAO_S  CTRSTR = SENSE_ERRMSG,-
                     0644  1011                 OUTLEN = BUFFER_PTR,-
                     0644  1012                 OUTBUF = FAO_BUF,-
                     0644  1013                 P1     = (R5),-
                     0644  1014                 P2     = 2(R5)
          000C'CF    DF  065C  1015         PUSHAL  BUFFER_PTR                 ; Error message
             0235    31  0660  1016         BRW     FAIL_OUT                   ; Failure exit
                     0663  1017
                     0663  1018 ERROR_TEST:
                     0663  1019         $SETSFM_S ENBFLG = #0                  ; Turn off system service mode
                     066C  1020
                     066C  1021 ;
                     066C  1022 ; Read data with IO$M_NOW specified but no data available
```

```
                     066C  1023 ;
                     066C  1024          $QIOW_S -                       ; Read data message
                     066C  1025                  CHAN = XD_CHAN,-
                     066C  1026                  FUNC = #IO$_READVBLK!IO$M_NOW,-
                     066C  1027                  IOSB = XD_IOSB,-
                     066C  1028                  P1 = RECV_BUF,-
                     066C  1029                  P2 = #128
                     0693  1030
     57  00000870 8F  D0  0693  1031          MOVL    #SS$_ENDOFFILE,R7
         58  0670'CF  3C  069A  1032          MOVZWL  XD_IOSB,R8
             57  58   B1  069F  1033          CMPW    R8,R7                   ; Correct error code?
                 3B   12  06A2  1034          BNEQ    ERRTST_ERR              ; Br if not
                      06A4  1035
                      06A4  1036 ;
                      06A4  1037 ; Buffer not enough to hold all information from IO$_SENSEMODE
                      06A4  1038 ;
                      06A4  1039
                      06A4  1040          $QIOW_S -                       ; Read device (trib. ) charracteristic
                      06A4  1041                  CHAN = XD_CHAN,-
                      06A4  1042                  FUNC = #IO$_SENSEMODE,-
                      06A4  1043                  IOSB = XD_IOSB,-
                      06A4  1044                  P1 = SENSE_P1BUF,-
                      06A4  1045                  P2 = #ERRTST_P2DESC
                      06CB  1046
     57  00000601 8F  D0  06CB  1047          MOVL    #SS$_BUFFEROVF,R7
         58  0670'CF  3C  06D2  1048          MOVZWL  XD_IOSB,R8
             57  58   B1  06D7  1049          CMPW    R8,R7                   ; Error code = buffer overflow?
                 03   12  06DA  1050          BNEQ    ERRTST_ERR              ; Error if not
                   0099  31  06DC  1051          BRW     READ_ERRCOUNT           ; Br to read and clear error count
                      06DF  1052
                      06DF  1053 ERRTST_ERR:
             04AF'CF  28  06DF  1054          MOVC3   COMP_STATUS_MSG,-       ; We need an error message...
0014'CF      04B7'CF      06E3  1055                  COMP_STATUS_MSG+8,BUFFER
             04AF'CF  A3  06E9  1056          SUBW3   COMP_STATUS_MSG,-       ; ...to compare...
     59      00FA 8F      06ED  1057                  #TEXT_BUFFER,R9
     000C'CF     59   3C  06F1  1058          MOVZWL  R9,BUFFER_PTR
     0010'CF     53   D0  06F6  1059          MOVL    R3,BUFFER_PTR+4
                      06FB  1060          $GETMSG_S MSGID = R7,-          ; ...the error we expected...
                      06FB  1061                  MSGLEN = BUFFER_PTR,-
                      06FB  1062                  BUFADR = BUFFER_PTR
0010'CF      000C'CF  C0  0710  1063          ADDL2   BUFFER_PTR,BUFFER_PTR+4
     59      000C'CF  A2  0717  1064          SUBW2   BUFFER_PTR,R9
             04E7'CF  28  071C  1065          MOVC3   RECEIVED_MSG,-
             04EF'CF      0720  1066                  RECEIVED_MSG+8,-
             0010'DF      0723  1067                  @BUFFER_PTR+4
     0010'CF     53   D0  0726  1068          MOVL    R3,BUFFER_PTR+4
     59      04E7'CF  A2  072B  1069          SUBW2   RECEIVED_MSG,R9
     000C'CF     59   3C  0730  1070          MOVZWL  R9,BUFFER_PTR
                      0735  1071          $GETMSG_S MSGID = R8,-          ; ...with the one we received
                      0735  1072                  MSGLEN = BUFFER_PTR,-
                      0735  1073                  BUFADR = BUFFER_PTR
     59      000C'CF  A2  074A  1074          SUBW2   BUFFER_PTR,R9
59   00FA 8F     59   A3  074F  1075          SUBW3   R9,#TEXT_BUFFER,R9
             59   59   3C  0755  1076          MOVZWL  R9,R9
0014'C9      2E22 8F  B0  0758  1077          MOVW    #^A/"./,BUFFER(R9)
000C'CF      59   02  A1  075F  1078          ADDW3   #2,R9,BUFFER_PTR
0010'CF      0014'CF  DE  0765  1079          MOVAL   BUFFER,BUFFER_PTR+4
```

```
                000C'CF    DF  076C  1080          PUSHAL  BUFFER_PTR                ; Error message
          02C6'CF    58    D0  0770  1081          MOVL    R8,STATUS                 ; Save our actual error as exit status
                0120    31  0775  1082          BRW     FAIL_OUT                  ; Failure exit
                            0778  1083
                            0778  1084 READ_ERRCOUNT:
                            0778  1085          $SETSFM_S ENBFLG = #1             ; Turn on  system service mode
                            0781  1086
                            0781  1087          $QIOW_S -                         ; Read and clear the error counters
                            0781  1088                  CHAN = XD_CHAN,-
                            0781  1089                  FUNC = #IO$_SENSEMODE!IO$M_RD_COUNT!IO$M_CLR_COUNT,-
                            0781  1090                  IOSB = XD_IOSB,-
                            0781  1091                  P2 = #ERRCOUNT_DESC
                            07A6  1092
                            07A6  1093          $FAO_S  CTRSTR = SENSE_PRM,-
                            07A6  1094                  OUTLEN = ALT_BUFFER_PTR,-
                            07A6  1095                  OUTBUF = ALT_FAO_BUF,-
                            07A6  1096                  P1     = #DEVDSC
            0116'CF    DF  07BF  1097          PUSHAL  ALT_BUFFER_PTR
          08A5'CF    01    FB  07C3  1098          CALLS   #1,CHECK_IOSB             ; Check status
                            07C8  1099
                            07C8  1100 CLEAN_EXIT:
                            07C8  1101
          0002'CF    20    AA  07C8  1102          BICW2   #FLAG_SHUTDNM,FLAG        ; Clear the shutdown flag
                            07CD  1103
                            07CD  1104          $QIOW_S -                         ; Shut down the device
                            07CD  1105                  CHAN = XD_CHAN,-
                            07CD  1106                  FUNC = #IO$_SETMODE!IO$M_CTRL!IO$M_SHUTDOWN,-
                            07CD  1107                  IOSB = XD_IOSB
                            07EE  1108
                            07EE  1109          $FAO_S  CTRSTR = SET_PRM,-
                            07EE  1110                  OUTLEN = ALT_BUFFER_PTR,-
                            07EE  1111                  OUTBUF = ALT_FAO_BUF,-
                            07EE  1112                  P1     = #DEVDSC
            0116'CF    DF  0807  1113          PUSHAL  ALT_BUFFER_PTR
          08A5'CF    01    FB  080B  1114          CALLS   #1,CHECK_IOSB             ; Check status
                            0810  1115
                            0810  1116 SUC_EXIT:
                            0810  1117          $TRNLOG_S LOGNAM = MODE,-
                            0810  1118                    RSLLEN = BUFFER_PTR,-
                            0810  1119                    RSLBUF = FAO_BUF        ; Get the run mode
          0014'CF    20    8A  0829  1120          BICB2   #LC_BITM,BUFFER           ; Convert to upper case
        0014'CF    4C  8F    91  082E  1121          CMPB    #^A/L/,BUFFER             ; Is this a loop for ever?
                    40    12  0834  1122          BNEQ    10$                       ; BR if not
          0002'CF    02    AA  0836  1123          BICW2   #TEST_OVERM,FLAG          ; Reset the termination flag
              02EA'CF    D6  083B  1124          INCL    PASS                      ; Bump the pass count
                            083F  1125          $FAO_S  CTRSTR = PASS_MSG,-
                            083F  1126                  OUTLEN = BUFFER_PTR,-
                            083F  1127                  OUTBUF = FAO_BUF,-
                            083F  1128                  P1     = PASS,-
                            083F  1129                  P2     = ITERATION,-
                            083F  1130                  P3     = #0               ; Make the end of pass message
                000C'CF    DF  085C  1131          PUSHAL  BUFFER_PTR                ; Push the string desc.
                      01    DD  0860  1132          PUSHL   #1                        ; Push arg count
             00741133 8F    DD  0862  1133          PUSHL   #UETP$_TEXT!STS$K_INFO    ; Push the signal name
        00000000'GF    03    FB  0868  1134          CALLS   #3,G^LIB$SIGNAL           ; Print the end of pass message
              02E6'CF    D4  086F  1135          CLRL    ITERATION                 ; Reset the iteration count
                 FBC7    31  0873  1136          BRW     RESTART                   ; Do the next pass
```

```
                     0876  1137 10$:
 56  0A88'CF  00000A88'8F  C1  0876  1138          ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; Set the unit block list header
                       02  88  0880  1139          BISB2   #UETUNT$M_TESTABLE,-
                       0B  A6  0882  1140                  UETUNT$B_FLAGS(R6)      ; Set the testable bit
     02C6'CF  10000001  8F  D0  0884  1141          MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,STATUS ; Set successful exit status
                              088D  1142          $EXIT_S STATUS                  ; Exit with the status
                              0898  1143
                              0898  1144 FAIL_OUT:                                ; Failure exit
                       01  DD  0898  1145          PUSHL   #1                              ; Arg count
                 00741132  8F  DD  089A  1146          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; Signal name
                       03  DD  08A0  1147          PUSHL   #3                              ; Arg count
                     02DC  31  08A2  1148          BRW     ERROR_EXIT                      ; Error exit
                              08A5  1149
```

```
                     08A5  1151          .SBTTL  CHECKIOSB - Check IO status block
                     08A5  1152 ;++
                     08A5  1153 ; FUNCTIONAL DESCRIPTION:
                     08A5  1154 ;       This routine checks the IO status block = #SS$_NORMAL
                     08A5  1155 ;
                     08A5  1156 ; CALLING SEQUENCE:
                     08A5  1157 ;       CALLS   #1,CHECK_IOSB
                     08A5  1158 ;
                     08A5  1159 ; INPUT PARAMETERS:
                     08A5  1160 ;       Address of error message
                     08A5  1161 ;
                     08A5  1162 ; IMPLICIT INPUTS:
                     08A5  1163 ;       XD_IOSB is the IOSB from some $QIO
                     08A5  1164 ;
                     08A5  1165 ; OUTPUT PARAMETERS:
                     08A5  1166 ;       NONE
                     08A5  1167 ;
                     08A5  1168 ; IMPLICIT OUTPUTS:
                     08A5  1169 ;       Exit with status if IOSB not right
                     08A5  1170 ;
                     08A5  1171 ; COMPLETION CODES:
                     08A5  1172 ;       IO status in STATUS if error
                     08A5  1173 ;
                     08A5  1174 ; SIDE EFFECTS:
                     08A5  1175 ;       Program exit if error found
                     08A5  1176 ;
                     08A5  1177 ;--
                     08A5  1178
                     08A5  1179 CHECK_IOSB:
                0004 08A5  1180          .WORD   ^M<R2>
01   0670'CF   B1    08A7  1181          CMPW    XD_IOSB,#SS$_NORMAL     ; Is the QIO O.K.?
          01   12    08AC  1182          BNEQ    10$                     ; Br if not
               04    08AE  1183          RET                             ; Return
                     08AF  1184 10$:
7E   0670'CF   3C    08AF  1185          MOVZWL  XD_IOSB,-(SP)           ; Push the error status code
02C6'CF   6E    D0   08B4  1186          MOVL    (SP),STATUS             ; Set return status
52   0411'CF   DE    08B9  1187          MOVAL   DMP_IOSB_DUMP,R2        ; Assume we're testing a DMP
          00'  91    08BE  1188          CMPB    S^#DT$_DMP11,-           ; But are we?
     0253'CF         08C0  1189                  DIBBUF+DIB$B_DEVTYPE
          05   13    08C3  1190          BEQL    20$
52   0393'CF   DE    08C5  1191          MOVAL   DMF_IOSB_DUMP,R2        ; Get a different string if not
                     08CA  1192 20$:
                     08CA  1193          $FAO_S  CTRSTR = (R2),-          ; Get the IOSB in plain text
                     08CA  1194                  OUTLEN = BUFFER_PTR,-
                     08CA  1195                  OUTBUF = FAO_BUF,-
                     08CA  1196                  P1     = @#XD_IOSB,-
                     08CA  1197                  P2     = @#XD_IOSB+2,-
                     08CA  1198                  P3     = @#XD_IOSB+4,-
                     08CA  1199                  P4     = @#XD_IOSB+5,-
                     08CA  1200                  P5     = @#XD_IOSB+6,-
                     08CA  1201                  P6     = @#XD_IOSB+7
     000C'CF    DF   08FF  1202          PUSHAL  BUFFER_PTR
          01    DD   0903  1203          PUSHL   #1
 00741132  8F    DD   0905  1204          PUSHL   #UETP$_TEXT!STS$K_ERROR
       04  AC    DD   090B  1205          PUSHL   04(AP)
          01    DD   090E  1206          PUSHL   #1
 00741132  8F    DD   0910  1207          PUSHL   #UETP$_TEXT!STS$K_ERROR
```

```
                07   DD  0916  1208          PUSHL   #7                      ; Argument count
            0266    31  0918  1209          BRW     ERROR_EXIT              ; Error exit
                        091B  1210
```

```
                     091B  1212          .SBTTL  Check QIO AST Routine
                     091B  1213 ;++
                     091B  1214 ; FUNCTIONAL DESCRIPTION:
                     091B  1215 ;       This routine will be called as a QIO completion AST routine
                     091B  1216 ;       It checks IO status block and the AST parameter
                     091B  1217 ;
                     091B  1218 ; CALLING SEQUENCE:
                     091B  1219 ;       Called via AST at $QIO completion
                     091B  1220 ;
                     091B  1221 ; INPUT PARAMETERS:
                     091B  1222 ;       NONE
                     091B  1223 ;
                     091B  1224 ; IMPLICIT INPUTS:
                     091B  1225 ;       DEVDSC, ALT_BUFFER_PTR, ALT_FAO_BUF and ALT_BUFFER used in forming a
                     091B  1226 ;               potential error message.
                     091B  1227 ;
                     091B  1228 ; OUTPUT PARAMETERS:
                     091B  1229 ;       NONE
                     091B  1230 ;
                     091B  1231 ; IMPLICIT OUTPUTS:
                     091B  1232 ;       Error message if error
                     091B  1233 ;
                     091B  1234 ; COMPLETION CODES:
                     091B  1235 ;       IO status in STATUS if error
                     091B  1236 ;
                     091B  1237 ; SIDE EFFECTS:
                     091B  1238 ;       Program exit if error
                     091B  1239 ;       BUFFER_PTR and BUFFER used if error
                     091B  1240 ;
                     091B  1241 ;--
                     091B  1242 CHK_QIO_AST:
                OFFC 091B  1243          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     091D  1244
                     091D  1245          $FAO_S  CTRSTR = @04(AP),-                  ; Form message for CHECK_IOSB
                     091D  1246                  OUTLEN = ALT_BUFFER_PTR,-
                     091D  1247                  OUTBUF = ALT_FAO_BUF,-
                     091D  1248                  P1     = #DEVDSC
          0116'CF    DF  0935  1249          PUSHAL  ALT_BUFFER_PTR
    FF67  CF  01     FB  0939  1250          CALLS   #1,CHECK_IOSB                   ; Go check IO status block
                         093E  1251          RET
```

```
                        093F  1253          .SBTTL  Receive data AST routine
                        093F  1254 ;++
                        093F  1255 ; FUNCTIONAL DESCRIPTION:
                        093F  1256 ;       This routine will be called as receive data AST routine
                        093F  1257 ;       It checks IO status and compare the data in the receive buffer
                        093F  1258 ;       against the transmit buffer
                        093F  1259 ;
                        093F  1260 ; CALLING SEQUENCE:
                        093F  1261 ;       Called via AST at $QIO READ
                        093F  1262 ;
                        093F  1263 ; INPUT PARAMETERS:
                        093F  1264 ;       AST parameter = message length
                        093F  1265 ;
                        093F  1266 ; IMPLICIT INPUTS:
                        093F  1267 ;       DEVDSC and various text buffers are used in forming error messages
                        093F  1268 ;
                        093F  1269 ; OUTPUT PARAMETERS:
                        093F  1270 ;       NONE
                        093F  1271 ;
                        093F  1272 ; IMPLICIT OUTPUTS:
                        093F  1273 ;       Error message if error found
                        093F  1274 ;
                        093F  1275 ; COMPLETION CODES:
                        093F  1276 ;       in STATUS
                        093F  1277 ;
                        093F  1278 ; SIDE EFFECTS:
                        093F  1279 ;       Program exit if error found
                        093F  1280 ;
                        093F  1281 ;--
                        093F  1282 RECV_AST:
                    0FFC  093F  1283          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
          01  0678'CF   B1  0941  1284          CMPW    RCV_IOSB,#SS$_NORMAL     ; Is the QIO O.K.?
                0C    12  0946  1285          BNEQ    10$                      ; Br if not
0680'CF 0880'CF 04 AC   29  0948  1286          CMPC3   4(AP),RECV_BUF,XMIT_BUF ; Compare the data
                2A    12  0951  1287          BNEQ    20$
                04    04  0953  1288          RET
                          0954  1289 10$:
                          0954  1290          $FAO_S  CTRSTR = READ_PRM,-
                          0954  1291                  OUTLEN = ALT_BUFFER_PTR,-
                          0954  1292                  OUTBUF = ALT_FAO_BUF,-
                          0954  1293                  P1     = #DEVDSC
    0670'CF 0678'CF   7D  096D  1294          MOVQ    RCV_IOSB,XD_IOSB         ; Set up a copy of our error status
            0116'CF   DF  0974  1295          PUSHAL  ALT_BUFFER_PTR
        FF28 CF   01  FB  0978  1296          CALLS   #1,CHECK_IOSB            ; Take advantage of existing routine
                          097D  1297          ; Note that we will not return!
                          097D  1298
                          097D  1299 20$:
            7E    61  9A  097D  1300          MOVZBL  (R1),-(SP)               ; Save the bad data...
            7E    63  9A  0980  1301          MOVZBL  (R3),-(SP)               ; ...the good data...
      7E    04 AC   50  C3  0983  1302          SUBL3   R0,4(AP),-(SP)           ; ...the offset of the mismatch...
         7E   02E2'CF  3C  0988  1303          MOVZWL  UNIT_NUMBER,-(SP)        ; ...the failing unit...
            0218'CF   7F  098D  1304          PUSHAQ  DEVDSC                   ; ...the device name...
                05    DD  0991  1305          PUSHL   #5                       ; ...and the count of parameters...
     00748012   8F    DD  0993  1306          PUSHL   #UETP$_DATAER!STS$K_ERROR ; ...for our error message
                07    DD  0999  1307          PUSHL   #7
              01E3    31  099B  1308          BRW     ERROR_EXIT
```

```
                     C99E  1310          .SBTTL  Half Minute Timer Expiration Routine
                     099E  1311 ;++
                     099E  1312 ; FUNCTIONAL DESCRIPTION:
                     099E  1313 ;       This routine will be called only if the timer which was set to prevent
                     099E  1314 ;       program hangs goes off.
                     099E  1315 ;
                     099E  1316 ; CALLING SEQUENCE:
                     099E  1317 ;       Called via AST at $SETIMR expiration.
                     099E  1318 ;
                     099E  1319 ; INPUT PARAMETERS:
                     099E  1320 ;       04(AP)  Address of a descriptor for an error message
                     099E  1321 ;
                     099E  1322 ; IMPLICIT INPUTS:
                     099E  1323 ;       DEVDSC and various text buffers are used to for error messages
                     099E  1324 ;
                     099E  1325 ; OUTPUT PARAMETERS:
                     099E  1326 ;       NONE
                     099E  1327 ;
                     099E  1328 ; IMPLICIT OUTPUTS:
                     099E  1329 ;       Time out error message
                     099E  1330 ;
                     099E  1331 ; COMPLETION CODES:
                     099E  1332 ;       NONE
                     099E  1333 ;
                     099E  1334 ; SIDE EFFECTS:
                     099E  1335 ;       Program exit
                     099E  1336 ;
                     099E  1337 ;--
                     099E  1338
                     099E  1339 TIME_ERR_OUT:
                0FFC 099E  1340          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     09A0  1341
                     09A0  1342          $FAO_S  CTRSTR = @04(AP),-
                     09A0  1343                  OUTLEN = ALT_BUFFER_PTR,-
                     09A0  1344                  OUTBUF = ALT_FAO_BUF,-
                     09A0  1345                  P1     = #DEVDSC
      0116'CF    DF 09B8  1346          PUSHAL  ALT_BUFFER_PTR          ; Set up our error message
            01   DD 09BC  1347          PUSHL   #1
00741132 8F    DD 09BE  1348          PUSHL   #UETP$_TEXT!STS$K_ERROR
            03   DD 09C4  1349          PUSHL   #3                      ; Push the argument count total
          01B8   31 09C6  1350          BRW     ERROR_EXIT              ; Bail out completely
```

```
                     09C9  1352          .SBTTL  Three Minutes Timer Expiration Routine
                     09C9  1353 ;++
                     09C9  1354 ; FUNCTIONAL DESCRIPTION:
                     09C9  1355 ;       This routine will be called when the device test has been run for
                     09C9  1356 ;       about three minutes. (that is, one normal run )
                     09C9  1357 ;
                     09C9  1358 ; CALLING SEQUENCE:
                     09C9  1359 ;       Called via AST at $SETIMR expiration.
                     09C9  1360 ;
                     09C9  1361 ; INPUT PARAMETERS:
                     09C9  1362 ;       NONE
                     09C9  1363 ;
                     09C9  1364 ; IMPLICIT INPUTS:
                     09C9  1365 ;       NONE
                     09C9  1366 ;
                     09C9  1367   OUTPUT PARAMETERS:
                     09C9  1368 ;       NONE
                     09C9  1369 ;
                     09C9  1370 ; IMPLICIT OUTPUTS:
                     09C9  1371 ;       NONE
                     09C9  1372 ;
                     09C9  1373 ; COMPLETION CODES:
                     09C9  1374 ;       NONE
                     09C9  1375 ;
                     09C9  1376 ; SIDE EFFECTS:
                     09C9  1377 ;       Get out the transmit /receive loop test
                     09C9  1378 ;
                     09C9  1379 ;--
                     09C9  1380
                     09C9  1381 TIME_SUC_OUT:
                OFFC 09C9  1382          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
0002'CF    02     A8 09CB  1383          BISW2   #TEST_OVERM,FLAG                    ; set test over bit
                   04 09D0  1384          RET
```

```
09D1  1386
09D1  1387          .SBTTL  System Service Exception Handler
09D1  1388 ;++
09D1  1389 ; FUNCTIONAL DESCRIPTION:
09D1  1390 ;       This routine is executed if a software or hardware exception occurs or
09D1  1391 ;       if a LIB$SIGNAL system service is used to output a message.
09D1  1392 ;
09D1  1393 ; CALLING SEQUENCE:
09D1  1394 ;       Entered via an exception from the system
09D1  1395 ;
09D1  1396 ; INPUT PARAMETERS:
09D1  1397 ;       ERROR_COUNT    = previous cumulative error count
09D1  1398 ;
09D1  1399 ;          AP ---->      +---------------+
09D1  1400 ;                        |       2       |
09D1  1401 ;                        +---------------+
09D1  1402 ;                        | SIGNL ARY PNT |
09D1  1403 ;                        +---------------+
09D1  1404 ;                        | MECH  ARY PNT |  ---------
09D1  1405 ;                        +---------------+          ^
09D1  1406 ;                        |       4       |          :
09D1  1407 ;                        +---------------+          :
09D1  1408 ;                        | ESTABLISH FP  |          :
09D1  1409 ;                        +---------------+          :
09D1  1410 ;                        |     DEPTH     |  Mechanism Array
09D1  1411 ;                        +---------------+          :
09D1  1412 ;                        |       R0      |          :
09D1  1413 ;                        +---------------+          :
09D1  1414 ;                        |       R1      |          v
09D1  1415 ;                        +---------------+  ---------
09D1  1416 ;                        |       N       |          ^
09D1  1417 ;                        +---------------+          :
09D1  1418 ;                        | CONDITION NAME|          :
09D1  1419 ;                        +---------------+          :
09D1  1420 ;                        | N-3 ADDITIONAL|     Signal Array
09D1  1421 ;                        | LONG WORD ARGS|          :
09D1  1422 ;                        +---------------+          :
09D1  1423 ;                        |       PC      |          :
09D1  1424 ;                        +---------------+          :
09D1  1425 ;                        |       PSL     |          v
09D1  1426 ;                        +---------------+  ---------
09D1  1427 ; IMPLICIT INPUTS:
09D1  1428 ;       NONE
09D1  1429 ;
09D1  1430 ; OUTPUT PARAMETERS:
09D1  1431 ;       NONE
09D1  1432 ;
09D1  1433 ; IMPLICIT OUTPUTS:
09D1  1434 ;       NONE
09D1  1435 ;
09D1  1436 ; COMPLETION CODES:
09D1  1437 ;       SS$_NORMAL if it's a UETP condition or RMS error.
09D1  1438 ;       Error status from exception, otherwise.
09D1  1439 ;
09D1  1440 ; SIDE EFFECTS:
09D1  1441 ;       May branch to ERROR_EXIT.
09D1  1442 ;       May print a message.
09D1  1443 ;--
```

```
                        09D1  1443
                        09D1  1444 SSERROR:
                   0FFC 09D1  1445          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        09D3  1446
                        09D3  1447          $SETAST_S ENBFLG = #0               ; Disable AST delivery
                01      DD  09D6  1448          PUSHL   #1                      ; Assume ASTs were enabled
             50 09      D1  09D8  1449          CMPL    S^#SS$_WASSET,R0        ; Were ASTs enabled?
                02      13  09DB  1450          BEQL    10$                     ; BR if they were
                6E      D4  09DD  1451          CLRL    (SP)                    ; Set ASTs to remain disabled
                        09DF  1452 10$:
                        09DF  1453          $SETSFM_S ENBFLG = #0               ; Disable SS failure mode
                01      DD  09E8  1454          PUSHL   #1                      ; Assume SS failure mode was enabled
             50 09      D1  09EA  1455          CMPL    S^#SS$_WASSET,R0        ; Was SS failure mode enabled?
                0?      13  09ED  1456          BEQL    20$                     ; BR if it was
                ?E      D4  09EF  1457          CLRL    (SP)                    ; Set SS failure mode to remain clr
                        09F1  1458 20$:
          56 04 AC      D0  09F1  1459          MOVL    CHF$L_SIGARGLST(AP),R6  ; Get the signal array pointer
          59 04 A6      7D  09F5  1460          MOVQ    CHF$L_SIG_NAME(R6),R9   ; Get NAME in R9 and ARG1 in R10
                10      ED  09F9  1461          CMPZV   #STS$V_FAC_NO,-         ; Is this a message from LIB$SIGNAL?
                0C          0A01  1462                  #STS$S_FAC_NO,-
00000074     8F 59          0A02  1463                  R9,#UETP$_FACILITY
                ?4      12  0A08  1464          BNEQ    30$                     ; BR if this is not a UETP exception
             66 02      C2  0A0A  1465          SUBL2   #2,CHF$L_SIG_ARGS(R6)   ; Drop the PC and PSL
                        0A0D  1466          $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
                21      11  0A1C  1467          BRB     40$                     ; Restore ASTs and SS fail mode
                        0A1E  1468 30$:
59   0000045C  8F      D1  0A1E  1469          CMPL    #SS$_SSFAIL,R9          ; RMS failures are SysSvc failures
                32      12  0A25  1470          BNEQ    50$                     ; BR if this can't be an RMS failure
                10      ED  0A27  1471          CMPZV   #STS$V_FAC_NO,-         ; Is it an RMS failure?
                0C          0A29  1472                  #STS$S_FAC_NO,-
             01 5A          0A2A  1473                  R10,#RMS$_FACILITY
                2B      12  0A2C  1474          BNEQ    50$                     ; BR if not
5A   F0000000  8F      CA  0A2E  1475          BICL2   #^XF0000000,R10         ; Strip control bits from status code
       08 A6    04      39  0A35  1476          MATCHC  #4,CHF$L_SIG_ARG1(R6),- ; Is it an RMS failure for which...
                14          0A39  1477                  #NRAT_LENGTH,-
          004D'CF           0A3A  1478                  NO_RMS_AST_TABLE        ; ...no AST can be delivered?
                1A      13  0A3D  1479          BEQL    50$                     ; BR if so - must give error here
                        0A3F  1480 40$:
                01      BA  0A3F  1481          POPR    #^M<R0>                 ; Restore SS failure mode...
                        0A41  1482          $SETSFM_S ENBFLG = R0               ; ...
                01      BA  0A4A  1483          POPR    #^M<R0>                 ; Restore AST enable...
                        0A4C  1484          $SETAST_S ENBFLG = R0               ; ...
             50 01      D0  0A55  1485          MOVL    S^#SS$_NORMAL,R0        ; Supply a standard status for exit
                04      04  0A58  1486          RET                             ; Resume processing (or goto RMS_ERROR)
                        0A59  1487 50$:
       02C6'CF  59      D0  0A59  1488          MOVL    R9,STATUS               ; Save the status
                58      D4  0A5F  1489          CLRL    R8                      ; Assume for now it's not SS failure
59   0000045C  8F      D1  0A60  1490          CMPL    #SS$_SSFAIL,R9          ; But is it a System Service failure?
                38      12  0A67  1491          BNEQ    70$                     ; BR if not - no special case message
                        0A69  1492          $GETMSG_S MSGID = R10,-             ; Get SS failure code associated text
                        0A69  1493                    MSGLEN = BUFFER_PTR,-
                        0A69  1494                    BUFADR = FAO_BUF,-
                        0A69  1495                    FLAGS  = #14,-
                        0A69  1496                    OUTADR = MSG_BLOCK
          02EF'CF   95      0A80  1497          TSTB    MSG_BLOCK+1             ; Get FAO arg count for SS failure code
                16      13  0A84  1498          BEQL    60$                     ; Don't use $GETMSG if no $FAO args...
          000C'CF   DF      0A86  1499          PUSHAL  BUFFER_PTR              ; ...else build up...
```

```
                01   DD  0A8A  1500          PUSHL   #1                              ; ...a message describing...
        00741130 8F   DD  0A8C  1501          PUSHL   #UETP$_TEXT                     ; ...why the System Service failed
              00 5A   F0  0A92  1502          INSV    R10,#STS$V_SEVERITY,-           ; Give the message...
              6E 03           0A95  1503                  #STS$S_SEVERITY,(SP)      ; ...the correct severity code
              58 03   D0  0A97  1504          MOVL    #3,R8                           ; Count the number of args we pushed
                 05   11  0A9A  1505          BRB     70$
                              0A9C  1506 60$:
                 5A   DD  0A9C  1507          PUSHL   R10                             ; Save SS failure code
              58 01   D0  0A9E  1508          MOVL    #1,R8                           ; Count the number of args we pushed
                              0AA1  1509 70$:
           57 66 04   C5  0AA1  1510          MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
              5E 57   C2  0AA5  1511          SUBL2   R7,SP                           ; Save the current signal array...
        6E 04 A6 57   28  0AA8  1512          MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
           7E 66 58   C1  0AAD  1513          ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
                 00CD 31  0AB1  1514          BRW     ERROR_EXIT
```

```
                     0AB4  1516          .SBTTL  RMS Error Handler
                     0AB4  1517 ;++
                     0AB4  1518 ; FUNCTIONAL DESCRIPTION:
                     0AB4  1519 ;       This routine handles error returns from RMS cal.s.
                     0AB4  1520 ;
                     0AB4  1521 ; CALLING SEQUENCE:
                     0AB4  1522 ;       Called by RMS when a file processing error is found.
                     0AB4  1523 ;
                     0AB4  1524 ; INPUT PARAMETERS:
                     0AB4  1525 ;       The FAB or RAB associated with the RMS call.
                     0AB4  1526 ;
                     0AB4  1527 ; IMPLICIT INPUTS:
                     0AB4  1528 ;       NONE
                     0AB4  1529 ;
                     0AB4  1530 ; OUTPUT PARAMETERS:
                     0AB4  1531 ;       NONE
                     0AB4  1532 ;
                     0AB4  1533 ; IMPLICIT OUTPUTS:
                     0AB4  1534 ;       Error message
                     0AB4  1535 ;
                     0AB4  1536 ; COMPLETION CODES:
                     0AB4  1537 ;       NONE
                     0AB4  1538 ;
                     0AB4  1539 ; SIDE EFFECTS:
                     0AB4  1540 ;       Program may exit, depending on severity of the error.
                     0AB4  1541 ;
                     0AB4  1542 ;--
                     0AB4  1543
                     0AB4  1544 RMS_ERROR:
                0FFC 0AB4  1545          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0AB6  1546
          56   04 AC   D0  0AB6  1547          MOVL    4(AP),R6                ; See whether we're dealing with...
             66   03   91  0ABA  1548          CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
                  16   12  0ABD  1549          BNEQ    10$                     ; BR if it's a RAB
       57   01FD'CF   DE  0ABF  1550          MOVAL   FILE,R7                 ; FAB-specific code:  text string...
             58   56   D0  0AC4  1551          MOVL    R6,R8                   ; ...address of FAB...
                0C A6   DD  0AC7  1552          PUSHL   FAB$L_STV(R6)           ; ...STV field for error...
                08 A6   DD  0ACA  1553          PUSHL   FAB$L_STS(R6)           ; ...STS field for error...
    02C6'CF     08 A6   D0  0ACD  1554          MOVL    FAB$L_STS(R6),STATUS    ; ...and save the error code
                  15   11  0AD3  1555          BRB     COMMON                  ; FAB and RAB share other code
                     0AD5  1556 10$:
       57   0209'CF   DE  0AD5  1557          MOVAL   RECORD,R7               ; RAB-specific code:  text string...
          58   3C A6   D0  0ADA  1558          MOVL    RAB$L_FAB(R6),R8        ; ...address of associated FAB...
                0C A6   DD  0ADE  1559          PUSHL   RAB$L_STV(R6)           ; ...STV field for error...
                08 A6   DD  0AE1  1560          PUSHL   RAB$L_STS(R6)           ; ...STS field for error...
    02C6'CF     08 A6   D0  0AE4  1561          MOVL    RAB$L_STS(R6),STATUS    ; ...and save the error code
                     0AEA  1562 COMMON:
          5A   34 A8   9A  0AEA  1563          MOVZBL  FAB$B_FNS(R8),R10       ; Get the file name size
                     0AEE  1564          $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                     0AEE  1565                  OUTLEN = BUFFER_PTR,-
                     0AEE  1566                  OUTBUF = FAO_BUF,-
                     0AEE  1567                  P1     = R7,-
                     0AEE  1568                  P2     = R10,-
                     0AEE  1569                  P3     = FAB$L_FNA(R8)
          000C'CF    DF  0B08  1570          PUSHAL  BUFFER_PTR              ; ...and arguments for ERROR_EXIT...
                01   DD  0B0C  1571          PUSHL   #1                      ; ...
       00741130 8F   DD  0B0E  1572          PUSHL   #UETP$_TEXT             ; ...
```

```
                00   EF  0B14  1573          EXTZV   #STS$V_SEVERITY,-
                03       0B16  1574                  #STS$S_SEVERITY,-
    59  02C6'CF          0B17  1575                  STATUS,R9                   ; ...get the severity code...
        6E    59    88    0B1B  1576          BISB2   R9,(SP)                     ; ...and add it into the signal name
              05    DD    0B1E  1577          PUSHL   #5                          ; Current arg count
                 005E   31  0B20  1578          BRW     ERROR_EXIT
```

```
                        0B23  1580          .SBTTL  CTRL/C Handler
                        0B23  1581 ;++
                        0B23  1582 ; FUNCTIONAL DESCRIPTION:
                        0B23  1583 ;       This routine handles CTRL/C AST's
                        0B23  1584 ;
                        0B23  1585 ; CALLING SEQUENCE:
                        0B23  1586 ;       Called via AST
                        0B23  1587 ;
                        0B23  1588 ; INPUT PARAMETERS:
                        0B23  1589 ;       NONE
                        0B23  1590 ;
                        0B23  1591 ; IMPLICIT INPUTS:
                        0B23  1592 ;       NONE
                        0B23  1593 ;
                        0B23  1594 ; OUTPUT PARAMETERS:
                        0B23  1595 ;       NONE
                        0B23  1596 ;
                        0B23  1597 ; IMPLICIT OUTPUTS:
                        0B23  1598 ;       NONE
                        0B23  1599 ;
                        0B23  1600 ; COMPLETION CODES:
                        0B23  1601 ;       NONE
                        0B23  1602 ;
                        0B23  1603 ; SIDE EFFECTS:
                        0B23  1604 ;       NONE
                        0B23  1605 ;
                        0B23  1606 ;--
                        0B23  1607
                        0B23  1608 CCASTHAND:
                  0FFC  0B23  1609          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0B25  1610
      21 0002'CF    05  E1  0B25  1611          BBC     #FLAG_SHUTDNV,FLAG,10$  ; Have to shut down device?
                        0B2B  1612          $QIO_S -                        ; Shut down the device
                        0B2B  1613                  CHAN = XD_CHAN,-
                        0B2B  1614                  FUNC = #IO$_SETMODE!IO$M_CTRL!IO$M_SHUTDOWN,-
                        0B2B  1615                  IOSB = XD_IOSB
                        0B4C  1616 10$:
          00A3'CF       DF  0B4C  1617          PUSHAL  CNTRLCMSG               ; Set message pointer
                01      DD  0B50  1618          PUSHL   #1                      ; Set arg count
        00741130        8F  DD  0B52  1619          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
                00      DD  0B58  1620          PUSHL   #0                      ; Indicate an abnormal termination
          0220'CF       DF  0B5A  1621          PUSHAL  PROCESS_NAME            ; ...
                02      DD  0B5E  1622          PUSHL   #2                      ; ...
        007410E0        8F  DD  0B60  1623          PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
  00000000'GF           07  FB  0B66  1624          CALLS   #7,G^LIB$SIGNAL         ; Output the message
                        D0  0B6D  1625          MOVL    #<STS$M_INHIB_MSG!-     ; Set the exit status
                        0B6E  1626                  SS$_CONTROLC-
                        0B6E  1627                  STS$K_SUCCESS+STS$K_WARNING>,-
  02C6'CF   10000650    8F      0B6E  1628                  STATUS
                        0B76  1629          $EXIT_S STATUS                  ; Terminate program cleanly
```

```
                        0B81  1631          .SBTTL  Error Exit
                        0B81  1632 ;++
                        0B81  1633 ; FUNCTIONAL DESCRIPTION:
                        0B81  1634 ;       This routine prints an error message and exits.
                        0B81  1635 ;
                        0B81  1636 ; CALLING SEQUENCE:
                        0B81  1637 ;       MOVx  error status value,STATUS
                        0B81  1638 ;       PUSHx error specific information on the stack
                        0B81  1639 ;       PUSHL current argument count
                        0B81  1640 ;       BRW   ERROR_EXIT
                        0B81  1641 ;
                        0B81  1642 ; INPUT PARAMETERS:
                        0B81  1643 ;       Arguments to LIB$SIGNAL, as above
                        0B81  1644 ;
                        0B81  1645 ; IMPLICIT INPUTS:
                        0B81  1646 ;       NONE
                        0B81  1647 ;
                        0B81  1648 ; OUTPUT PARAMETERS:
                        0B81  1649 ;       Message to SYS$OUTPUT and SYS$ERROR
                        0B81  1650 ;
                        0B81  1651 ; IMPLICIT OUTPUTS:
                        0B81  1652 ;       Program exit
                        0B81  1653 ;
                        0B81  1654 ; COMPLETION CODES:
                        0B81  1655 ;       Error in STATUS
                        0B81  1656 ;
                        0B81  1657 ; SIDE EFFECTS:
                        0B81  1658 ;       NONE
                        0B81  1659 ;
                        0B81  1660 ;--
                        0B81  1661
                        0B81  1662 ERROR_EXIT:
                        0B81  1663
                        0B81  1664          $SETAST_S ENBFLG = #0              ; ASTs can play havoc with messages
      15 0002'CF    03    E0  0B8A  1665          BBS     #BEGIN_MSGV,FLAG,10$      ; BR if "begin" msg already printed
                     7E    D4  0B90  1666          CLRL    -(SP)                     ; Set the time stamp flag
            000F'CF    DF    0B92  1667          PUSHAL  TEST_NAME                 ; Set the test name
                     02    DD  0B96  1668          PUSHL   #2                        ; Push the argument count
          00741039 8F    DD  0B98  1669          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
    00000000'GF    04    FB  0B9E  1670          CALLS   #4,G^LIB$SIGNAL           ; Print the startup message
                        0BA5  1671 10$:
    0302'CF    08    8E    C1  0BA5  1672          ADDL3   (SP)+,#8,ARG_COUNT        ; Get total # args, pop partial count
            02C2'CF    00    D6  0BAB  1673          INCL    ERROR_COUNT               ; Keep running error count
                     00    DD  0BAF  1674          PUSHL   #0                        ; Push the time parameter
            0220'CF    DF    0BB1  1675          PUSHAL  PROCESS_NAME              ; Push test name...
          000F0002 8F    DD  0BB5  1676          PUSHL   #^XF0002                  ; ...arg count...
          007410E2 8F    DD  0BBB  1677          PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
            02C2'CF    DD  0BC1  1678          PUSHL   ERROR_COUNT               ; Finish off arg list...
            0220'CF    DF  0BC5  1679          PUSHAL  PROCESS_NAME              ; ...
          00010002 8F    DD  0BC9  1680          PUSHL   #^X10002                  ; ...
          00748022 8F    DD  0BCF  1681          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
00000000'GF    0302'CF    FB  0BD5  1682          CALLS   ARG_COUNT,G^LIB$SIGNAL    ; Truly bitch
                        0BDE  1683
            02C6'CF    D5  0BDE  1684          TSTL    STATUS                    ; Did we exit with an error code?
                     09    12  0BE2  1685          BNEQ    20$                       ; BR if we did
          007410E2 8F    D0  0BE4  1686          MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
            02C6'CF        0BEA  1687                  STATUS
```

```
                     0BED  1688 20$:
                     0BED  1689          BBC     #FLAG_SHUTDNV,FLAG,30$  ; Have to shut down device?
                     0BF3  1690          $QIO_S -                        ; Shut down the device
                     0BF3  1691                  CHAN = XD_CHAN,-
                     0BF3  1692                  FUNC = #IO$_SETMODE!IO$M_CTRL!IO$M_SHUTDOWN
                     0C12  1693 30$:
                     0C12  1694          BISL    #STS$M_INHIB_MSG,STATUS ; Don't print messages twice!
                     0C1B  1695          $EXIT_S STATUS                  ; Exit in error
```

```
                     0C26  1697          .SBTTL  Exit Handler
                     0C26  1698 ;++
                     0C26  1699 ; FUNCTIONAL DESCRIPTION:
                     0C26  1700 ;       This routine handles cleanup at exit.  If the MODE logical name is
                     0C26  1701 ;       equated to "ONE", the routine will update the test flag in the
                     0C26  1702 ;       UETINIDEV.DAT file depending on the UETUNT$M_TESTABLE flag state in the
                     0C26  1703 ;       UETUNT$B_FLAGS field of the unit block for each unit for the device
                     0C26  1704 ;       under test.
                     0C26  1705 ;
                     0C26  1706 ; CALLING SEQUENCE:
                     0C26  1707 ;       Invoked automatically by $EXIT System Service.
                     0C26  1708 ;
                     0C26  1709 ; INPUT PARAMETERS:
                     0C26  1710 ;       STATUS  contains the exit status.
                     0C26  1711 ;       FLAG    has synchronizing bits.
                     0C26  1712 ;       DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
                     0C26  1713 ;
                     0C26  1714 ; IMPLICIT INPUTS:
                     0C26  1715 ;       UNIT_LIST points to the head of a doubly linked circular list of unit
                     0C26  1716 ;                 blocks for the device under test.
                     0C26  1717 ;
                     0C26  1718 ; OUTPUT PARAMETERS:
                     0C26  1719 ;       NONE
                     0C26  1720 ;
                     0C26  1721 ; IMPLICIT OUTPUTS:
                     0C26  1722 ;       Various files are de-accessed, the process name is reset, and any
                     0C26  1723 ;       necessary synchronization with UETPDEV01 is carried out.
                     0C26  1724 ;       If the MODE logical name is equated to "ONE", the routine will update
                     0C26  1725 ;       the test flag in the UETINIDEV.DAT file depending on the
                     0C26  1726 ;       UETUNT$M_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
                     0C26  1727 ;       block for each unit for the device under test.
                     0C26  1728 ;
                     0C26  1729 ; COMPLETION CODES:
                     0C26  1730 ;       NONE
                     0C26  1731 ;
                     0C26  1732 ; SIDE EFFECTS:
                     0C26  1733 ;       NONE
                     0C26  1734 ;
                     0C26  1735 ;--
                     0C26  1736
                     0C26  1737 EXIT_HANDLER:
                 0FFC  0C26  1738          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0C28  1739
                     0C28  1740          $SETSFM_S ENBFLG = #0               ; Turn off System Service failure mode
                     0C31  1741          $SETAST_S ENBFLG = #0               ; No more ASTs
                     0C3A  1742          $TRNLOG_S LOGNAM = MODE,-           ; Get the run mode
                     0C3A  1743                    RSLLEN = BUFFER_PTR,-
                     0C3A  1744                    RSLBUF = FAO_BUF
   0014'CF    20   8A  0C53  1745          BICB2   #LC_BITM,BUFFER             ; Convert to upper case
0014'CF    4F 8F   91  0C58  1746          CMPB    #^A/O/,BUFFER               ; Is this a one shot?
             03   13  0C5E  1747          BEQL    10$                         ; BR if yes...
           00B8   31  0C60  1748          BRW     END_UPDATE                  ; ...else don't update UETINIDEV.DAT
                     0C63  1749 10$:
03 0002'CF    02   E0  0C63  1750          BBS     #SAFE_TO_UPDV,FLAG,20$      ; Only update if it's safe
           00AF   31  0C69  1751          BRW     END_UPDATE                  ; Else forget it
                     0C6C  1752 20$:
   5A   0B7C'CF    DE  0C6C  1753          MOVAL   INI_RAB,R10                 ; Set the RAB address
```

```
                1E AA    02    90  0C71  1754          MOVB    #RAB$C_RFA,RAB$B_RAC(R10) ; Set RFA mode
          10 AA  0BC0'CF   06    28  0C75  1755          MOVC3   #6,DDB_RFA,RAB$W_RFA(R10) ; Set RFA to DDB line
                                     0C7C  1756          $GET    RAB = (R10)               ; Go back to the DDB record
                   75 50    E9    0C85  1757          BLBC    R0,UPDATE_FAILED          ; If failure then forget it
                1E AA    00    90  0C88  1758          MOVB    #RAB$C_SEQ,RAB$B_RAC(R10) ; Set back to sequential mode
     5B  0A88'CF  00000A88'8F    C1  0C8C  1759          ADDL3   #UNIT_LIST,UNIT_LIST,R11 ; Set the unit block list header
                      59    D4  0C96  1760          CLRL    R9                        ; Init a counter
                                     0C98  1761 UNIT_LOOP:
                      01    E1  0C98  1762          BBC     #UETUNT$V_TESTABLE,-      ; BR if this unit is not testable
                   02 0B AB         0C9A  1763                  UETUNT$B_FLAGS(R11),10$
                      59    D6  0C9D  1764          INCL    R9                        ; Count testable units
                                     0C9F  1765 10$:
                   5B    6B    C0  0C9F  1766          ADDL2   (R11),R11                 ; Next unit block
          00000A88'8F    5B    D1  0CA2  1767          CMPL    R11,#UNIT_LIST            ; Are we full circle in the list?
                         ED    12  0CA9  1768          BNEQ    UNIT_LOOP                 ; BR if not
                      59    D5  0CAB  1769          TSTL    R9                        ; Any testable units?
                      12    12  0CAD  1770          BNEQ    20$                       ; BR if yes...
           0018'CF    4E    8F  90  0CAF  1771          MOVB    #^A/N/,BUFFER+4           ; ...else disable the DDB record...
                                     0CB5  1772          $UPDATE RAB = (R10)               ; ...here
                   3C 50    E9    0CBE  1773          BLBC    R0,UPDATE_FAILED          ; If error then forget it
                                     0CC1  1774 20$:
                   5B    6B    C0  0CC1  1775          ADDL2   (R11),R11                 ; Next unit block
          00000A88'8F    5B    D1  0CC4  1776          CMPL    R11,#UNIT_LIST            ; Are we full circle in the list?
                         4E    13  0CCB  1777          BEQL    END_UPDATE                ; BR if yes
                                     0CCD  1778          $GET    RAB = (R10)               ; Get a record
                   24 50    E9    0CD6  1779          BLBC    R0,UPDATE_FAILED          ; If error then forget it
              0014'CF    20    8A  0CD9  1780          BICB2   #LC_BITM,BUFFER           ; Convert to uppercase
           0014'CF    55    8F  91  0CDE  1781          CMPB    #^A7U/,BUFFER             ; Is it a UCB record?
                         35    12  0CE4  1782          BNEQ    END_UPDATE                ; BR if not
                      01    E0  0CE6  1783          BBS     #UETUNT$V_TESTABLE,-      ; BR if this unit is testable...
                   D6 0B AB         0CE8  1784                  UETUNT$B_FLAGS(R11),20$
           0018'CF    4E    8F  90  0CEB  1785          MOVB    #^A/N/,BUFFER+4           ; ...else disable the UCB record...
                                     0CF1  1786          $UPDATE RAB = (R10)               ; ...here
                   C4 50    E8    0CFA  1787          BLBS    R0,20$                    ; Look at the next record if no error
                                     0CFD  1788 UPDATE_FAILED:
                   0C AA    DD  0CFD  1789          PUSHL   RAB$L_STV(R10)            ; Do a simple message...
                      50    DD  0D00  1790          PUSHL   R0                        ; ...to tell of the failure
                01B8'CF    DF  0D02  1791          PUSHAL  INIDEV_UPDERR
                      01    DD  0D06  1792          PUSHL   #1
                      00    EF  0D08  1793          EXTZV   #STS$V_SEVERITY,-         ; Copy the severity from RMS status...
          7E    50    03         0D0A  1794                  #STS$S_SEVERITY,R0,-(SP)
     6E  00741130    8F    C8  0D0D  1795          BISL2   #UETP$_TEXT,(SP)          ; ...to our message
00000000'GF         05    FB  0D14  1796          CALLS   #5,G^LIB$SIGNAL
                                     0D1B  1797 END_UPDATE:
                      00    DD  0D1B  1798          PUSHL   #0                        ; Set the time flag
                000F'CF    DF  0D1D  1799          PUSHAL  TEST_NAME                 ; Push the test name
                      02    DD  0D21  1800          PUSHL   #2                        ; Push arg count
                      00    EF  0D23  1801          EXTZV   #STS$V_SEVERITY,-         ; Push the proper exit severity...
                      03         0D25  1802                  #STS$S_SEVERITY,-
          7E  02C6'CF          0D26  1803                  STATUS,-(SP)
     6E  00741080    8F    C8  0D2A  1804          BISL2   #UETP$_ENDEDD,(SP)        ; ...and use it in our message code
                      04    DD  0D31  1805          PUSHL   #4
                   51 5E    D0  0D33  1806          MOVL    SP,R1
                                     0D36  1807          $PUTMSG_S MSGVEC = (R1)           ; Output the message
                                     0D45  1808          $SETPRN_S PRCNAM = ACNT_NAME      ; Reset the process name
                      04    0D50  1809          RET                               ; That's all folks!
                                     0D51  1810
```

```
                0D51  1811          .END    UETDMPF00
```

```
$$.TAB                  = 00000C68 R      03
$$.TABEND               = 00000CAC R      03
$$.TMP                  = 00000000
$$.TMP1                 = 00000001
$$.TMP2                 = 0000006A
$$.TMPX                 = 00000016 R      04
$$.TMPX1                = 0000000D
$$T1                    = 00000001
$$T2                    = 00000006
ACNT_NAME                 00000000 R      02
ALL_SET                   000003E6 R      05
ALT_BUFFER                0000011E R      03
ALT_BUFFER_PTR            00000116 R      03
ALT_FAO_BUF               0000010E R      03
ARG_COUNT                 00000302 R      03
BAD_DATA                  00000A80 R      03
BEGIN_MSGM              = 00000008
BEGIN_MSGV              = 00000003
BUFFER                    00000014 R      03
BUFFER_PTR                0000000C R      03
BUF_DESC                  0000030A R      03
BUF_LEN                   00000308 R      03
CCASTHAND                 00000B23 R      05
CHAN_BUF                  00000312 R      03
CHECK_IOSB                000008A5 R      05
CHF$L_SIGARGLST         = 00000004
CHF$L_SIG_ARG1          = 00000008
CHF$L_SIG_ARGS          = 00000000
CHF$L_SIG_NAME          = 00000004
CHK_QIO_AST               0000091B R      05
CLEAN_EXIT                000007C8 R      05
CNTRL_MSG                 000000A3 R      02
COMMON                    00000AEA R      05
COMP_STATUS_MSG           000004AF R      02
CONTROLLER                00000031 R      02
CONT_DESC                 000001F5 R      02
CS1                       00000082 R      02
CS3                       00000094 R      02
DDB_RFA                   00000BC0 R      03
DEAD_CTRLNAME             000000E4 R      02
DEV$V_TRM               = 00000002
DEVDEP_SIZE             = 00000000
DEVDSC                    00000218 R      03
DEVNAM_LEN                000002E4 R      03
DEV_NAME                  00000237 R      03
DIB                       00000246 R      03
DIB$B_DEVCLASS          = 00000004
DIB$B_DEVTYPE           = 00000005
DIB$K_LENGTH            = 00000074
DIBBUF                    0000024E R      03
DMF_IOSB_DUMP             00000393 R      02
DMP_IOSB_DUMP             00000411 R      02
DTS_DMP1T                 ********   X    05
DUMMY_FAB                 00000C18 R      03
DUMMY_RAB                 00000C68 R      03
DVI$_DEVNAM             = 00000020
EFN2                    = 00000004
END_UPDATE                00000D1B R      05
ERRCNT_BUF                00000470 R      03
ERRCNT_LEN              = 00000200
ERRCOUNT_DESC             00000468 R      03
ERROR_COUNT               000002C2 R      03
ERROR_EXIT                00000B81 R      05
ERROR_TEST                00000663 R      05
ERRTEST_MSG               000005C0 R      02
ERRTST_ERR                000006DF R      05
ERRTST_P2BUF              00000460 R      03
ERRTST_P2DESC             00000458 R      03
ERRTST_P2LEN            = 00000008
ESC                     = 0000001B
EXIT_DESC                 000002F2 R      03
EXIT_HANDLER              00000C26 R      05
FAB$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$W_GBC               = 00000048
FAIL_OUT                  00000898 R      05
FAO_BUF                   00000004 R      03
FILE                      000001FD R      02
FIND_IT                   000001E1 R      05
FLAG                      00000002 R      03
FLAG_SHUTDNM            = 00000020
FLAG_SHUTDNV            = 00000005
FOUND_IT                  00000279 R      05
GOOD_DATA                 00000A81 R      03
HALFMIN                   000001E5 R      02
ILLEGAL_REC               00000151 R      02
INADDRESS                 000002D2 R      03
INIDEV_UPDERR             000001B8 R      02
INI_FAB                   00000B2C R      03
INI_RAB                   00000B7C R      03
INPUT_ITMLST              00000072 R      02
IO$M_CLR_COUNT            ********     X  05
IO$M_CTRL                 ********     X  05
IO$M_CTRLCAST             ********     X  05
IO$M_NOW                  ********     X  05
```

```
IO$M_RD_COUNT                   ********   X   05
IO$M_SHUTDOWN                   ********   X   05
IO$M_STARTUP                    ********   X   05
IO$_READVBLK                    ********   X   05
IO$_SENSEMODE                   ********   X   05
IO$_SETMODE                     ********   X   05
IO$_WRITEVBLK                   ********   X   05
ITERATION                       000002E6 R     03
LC_BITM                       = 00000020
LIB$SIGNAL                      ********   X   05
LIMIT                         = 00000010
LOOPBACK_TEST                   0000051D R     05
MAX_DEV_DESIG                 = 0000000A
MAX_MSG_LEN                   = 00000200
MAX_PROC_NAME                 = 0000000F
MAX_UNIT_DESIG                = 00000005
MODE                            00000041 R     02
MODE_IS_ONEM                  = 00000010
MODE_IS_ONEV                  = 00000004
MSG_BLOCK                       000002EE R     03
NAME_LEN                      = 0000000F
NEW_NODE                        00000A90 R     03
NMA$C_LINCN_LOO               = 00000001
NMA$C_LINPR_POI               = 00000000
NMA$C_PCCI_TRI                = 00000474
NMA$C_PCLI_CON                = 00000456
NMA$C_PCLI_PRO                = 00000458
NOUNIT_SELECTED                 0000012B R     02
NO_CTRLNAME                     000000C4 R     02
NO_RMS_AST_TABLE                0000004D R     02
NRAT_LENGTH                   = 00000014
OTS$CVT_TI_L                    ********   X   05
OUTADDRESS                      000002DA R     03
P1BUF                           00000386 R     03
P2BUF                           0000039E R     03
P2BUF_DESC                      00000396 R     03
P2BUF_LEN                     = 0000000C
PAGES                         = 00000001
PASS                            000002EA R     03
PASS_MSG                        00000185 R     02
PMTSIZ                        = 00000019
PRM                           = 00000064
PROCESS_NAME                    00000220 R     03
PROCESS_NAME_FREE             = 0000000B
PROC_CONT_NAME                  0000008B R     05
PROMPT                          00000238 R     02
QUAD_STATUS                     000002CA 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$W_RSZ                     = 00000022
RCV_IOSB                        00000678 R     03
READ_ERRCOUNT                   00000778 R     05
READ_PRM                        0000031A R     02
READ_SIZE                     = 00000000
RECEIVED_MSG                    000004E7 R     02
RECORD                          00000209 R     02
RECV                            00000581 R     05
RECVPOOL_SIZ                  = 00000004
RECV_AST                        0000093F R     05
RECV_BUF                        00000880 R     03
RECV_EFN                      = 00000008
RECV_ERR_MSG                    000004FF R     02
REC_SIZE                      = 00000028
RESTART                         0000043D R     05
RMS$_BLN                        ********     X 02
RMS$_BUSY                       ********     X 02
RMS$_CDA                        ********     X 02
RMS$_FAB                        ********     X 02
RMS$_FACILITY                 = 00000001
RMS$_RAB                        ********     X 02
RMS_ERROR                       00000AB4 R     05
RMS_ERR_STRING                  00000217 R     02
RW_TIME_ID                    = 00000003
RW_TO_MSG                       00000275 R     02
SAFE_TO_UPDM                  = 00000004
SAFE_TO_UPDV                  = 00000002
SEC$M_EXPREG                    ********   X   05
SEC$M_GBL                       ********   X   05
SENSE_ERRMSG                    00000541 R     02
SENSE_P1BUF                     000003B8 R     03
SENSE_P2BUF                     000003C8 R     03
SENSE_P2DESC                    000003C0 R     03
SENSE_P2LEN                   = 00000090
SENSE_PRM                       00000339 R     02
SENSE_TEST                      000005D6 R     05
SET_PRM                         00000366 R     02
SET_XMIT_BUF                    0000052B R     05
SHR$_ABENDD                   = 000010E0
SHR$_BEGIND                   = 00001038
SHR$_ENDEDD                   = 00001080
SHR$_OPENIN                   = 00001098
SHR$_TEXT                     = 00001130
SS$_BADPARAM                  = 00000014
SS$_BUFFEROVF                 = 00000601
SS$_CONTROLC                  = 00000651
SS$_ENDOFFILE                 = 00000870
SS$_NORMAL                    = 00000001
SS$_NOSUCHSEC                 = 00000978
SS$_SSFAIL                    = 0000045C
SS$_WASSET                    = 00000009
SSERROR                         000009D1 R     05
SS_SYNCH_EFN                  = 00000003
```

```
START_CONT                      00000461 R      05
START_CONT_PRM                  000002A2 R      02
START_TEST                      00000407 R      05
START_TO_MSG                    00000251 R      02
START_TRI                       00000492 R      05
START_TRIB_PRM                  000002D0 R      02
STATUS                          000002C6 R      03
STR$UPCASE                      ******** X      05
STS$K_ERROR                   = 00000002
STS$K_INFO                    = 00000003
STS$K_SUCCESS                 = 00000001
STS$K_WARNING                 = 00000000
STS$M_INHIB_MSG               = 10000000
STS$S_FAC_NO                  = 0000000C
STS$S_SEVERITY                = 00000003
STS$V_FAC_NO                  = 00000010
STS$V_SEVERITY                = 00000000
SUC_EXIT                        00000810 R      05
SUPDEV_GBLSEC                   00000020 R      02
SUP_FAB                         00000BC8 R      03
SYS$ASSIGN                      ******** GX     05
SYS$CANTIM                      ******** GX     05
SYS$CONNECT                     ******** GX     05
SYS$CRMPSC                      ******** GX     05
SYS$DCLEXH                      ******** GX     05
SYS$EXIT                        ******** GX     05
SYS$EXPREG                      ******** GX     05
SYS$FAO                         ******** X      05
SYS$GET                         ******** GX     05
SYS$GETDEV                      ******** GX     05
SYS$GETDVI                      ******** GX     05
SYS$GETMSG                      ******** GX     05
SYS$INPUT                       00000061 R      02
SYS$MGBLSC                      ******** GX     05
SYS$OPEN                        ******** GX     05
SYS$PUTMSG                      ******** GX     05
SYS$QIO                         ******** GX     05
SYS$QIOW                        ******** GX     05
SYS$SETAST                      ******** GX     05
SYS$SETIMR                      ******** GX     05
SYS$SETPRN                      ******** GX     05
SYS$SETSFM                      ******** GX     05
SYS$TRNLOG                      ******** GX     05
SYS$UPDATE                      ******** GX     05
SYS$WAITFR                      ******** GX     05
SYSIN_FAB                       00000A98 R      03
SYSIN_RAB                       00000AE8 R      03
TEST_NAME                       0000000F R      02
TEST_OVERM                    = 00000002
TEST_OVERV                    = 00000001
TEXT_BUFFER                   = 000000FA
THREEMIN                        000001DD R      02
TIME_ERR_OUT                    0000099E R      05
TIME_ID_T                     = 00000001
TIME_SUC_OUT                    000009C9 R      05
TR_P1BUF                        0000038E R      03
TR_P2BUF                        000003B2 R      03
TR_P2BUF_DESC                   000003AA R      03
TR_P2BUF_LEN                  = 00000006
TTCHAN                          00000000 R      03
UETDMPF00                       00000000 RG     05
UETP                          = 00740000
UETP$_ABENDD                  = 007410E0
UETP$_ABORTC                  = 0074832B
UETP$_BEGIND                  = 00741038
UETP$_DATAER                  = 00748010
UETP$_DENOSU                  = 00748333
UETP$_DEUNUS                  = 0074819A
UETP$_ENDEDD                  = 00741080
UETP$_ERBOXPROC               = 00748020
UETP$_FACILITY                = 00000074
UETP$_OPENIN                  = 00741098
UETP$_TEXT                    = 00741130
UETUNT$B_FLAGS                = 0000000B
UETUNT$B_TYPE                 = 00000008
UETUNT$C_FAB                  = 00000110
UETUNT$C_INDSIZ               = 000001A4
UETUNT$K_FAB                  = 00000110
UETUNT$K_RAB                  = 00000160
UETUNT$M_TESTABLE             = 00000002
UETUNT$T_FILSPC               = 00000014
UETUNT$V_TESTABLE             = 00000001
UETUNT$W_SIZE                 = 00000009
UNIT_DESC                       000001ED R      02
UNIT_LIST                       00000A88 R      03
UNIT_LOOP                       00000C98 R      05
UNIT_NUMBER                     000002E2 R      03
UPDATE_FAILED                   00000CFD R      05
WRITE_PRM                       000002FD R      02
WRITE_SIZE                    = 00000000
XD_CHAN                         00000306 R      03
XD_IOSB                         00000670 R      03
XM$M_CHR_LOOPB                = 00000002
XMIT                            00000556 R      05
XMIT_BUF                        00000680 R      03
XMIT_EFN                      = 00000005
```

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	000005DC (1500.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RWDATA	00000CAC (3244.)	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
DMPF	00000D51 (3409.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	40	00:00:00.07	00:00:00.58
Command processing	141	00:00:00.67	00:00:05.85
Pass 1	1114	00:00:30.70	00:01:12.41
Symbol table sort	9	00:00:03.39	00:00:07.81
Pass 2	472	00:00:07.37	00:00:16.21
Symbol table output	39	00:00:00.30	00:00:01.08
Psect synopsis output	4	00:00:00.04	00:00:00.06
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1822	00:00:42.55	00:01:44.02

The working set limit was 2000 pages.
171259 bytes (335 pages) of virtual memory were used to buffer the intermediate code.
There were 130 pages of symbol table space allocated to hold 2362 non-local and 39 local symbols.
1811 source lines were read in Pass 1, producing 41 object records in Pass 2.
59 pages of virtual memory were used to define 52 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	46
TOTALS (all libraries)	49

2653 GETS were required to define 49 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:UETDMPF00/OBJ=OBJ$:UETDMPF00 MSRC$:UETDMPF00/UPDATE=(ENH$:UETDMPF00)+EXECML$/LIB+LIB$:UETP/LIB
```

SATSSF18
LIS

SATSSS09
LIS

UETCOMS00
LIS

UETDISK00
LIS

SATSSS10
LIS

UETQMPF00
LIS

UETFORT03
LIS

UETFORT01
LIS

UETLPAK00
LIS

UETNETS00
LIS

UETDRIW00
LIS

UETFORT02
LIS