RRRRRRRRRRR TTTTTTTTTTTTTTT PPPPPPPPPPP AAAAAAAAA DDDDDDDDDD
RRRRRRRRRRR TTTTTTTTTTTTTTT PPPPPPPPPPP AAAAAAAAA DDDDDDDDDD
RRRRRRRRRRR TTTTTTTTTTTTTTT PPPPPPPPPPP AAAAAAAAA DDDDDDDDDD
RRR RRR TTT PPP PPP AAA AAA DDD DDD
RRR RRR TTT PPP PPP AAA AAA DDD DDD
RRR RRR TTT PPP PPP AAA AAA DDD DDD
RRR RRR TTT PPP PPP AAA AAA DDD DDD
RRR RRR TTT PPP PPP AAA AAA DDD DDD
RRRRRRRRRRR TTT PPPPPPPPPPP AAA AAA DDD DDD
RRRRRRRRRRR TTT PPPPPPPPPPP AAA AAA DDD DDD
RRRRRRRRRRR TTT PPPPPPPPPPP AAA AAA DDD DDD
RRR RRR TTT PPP AAAAAAAAAAAAAAA DDD DDD
RRR RRR TTT PPP AAAAAAAAAAAAAAA DDD DDD
RRR RRR TTT PPP AAAAAAAAAAAAAAA DDD DDD
RRR RRR TTT PPP AAA AAA DDD DDD
RRR RRR TTT PPP AAA AAA DDD DDD
RRR RRR TTT PPP AAA AAA DDD DDD
RRR RRR TTT PPP AAA AAA DDDDDDDDDDD
RRR RRR TTT PPP AAA AAA DDDDDDDDDDD
RRR RRR TTT PPP AAA AAA DDDDDDDDDDD

FILEID**CTERMRT

```
  CCCCCCCC  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR    MM      MM  RRRRRRRR    TTTTTTTTTT
  CCCCCCCC  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR    MM      MM  RRRRRRRR    TTTTTTTTTT
CC              TT      EE          RR      RR  MMMM  MMMM  RR      RR      TT
CC              TT      EE          RR      RR  MMMM  MMMM  RR      RR      TT
CC              TT      EE          RR      RR  MM  MM  MM  RR      RR      TT
CC              TT      EE          RR      RR  MM  MM  MM  RR      RR      TT
CC              TT      EEEEEEEE    RRRRRRRR    MM      MM  RRRRRRRR        TT
CC              TT      EEEEEEEE    RRRRRRRR    MM      MM  RRRRRRRR        TT
CC              TT      EE          RR  RR      MM      MM  RR  RR          TT
CC              TT      EE          RR  RR      MM      MM  RR  RR          TT
CC              TT      EE          RR    RR    MM      MM  RR    RR        TT          ....
CC              TT      EE          RR    RR    MM      MM  RR    RR        TT          ....
  CCCCCCCC      TT      EEEEEEEEEE  RR      RR  MM      MM  RR      RR      TT          ....
  CCCCCCCC      TT      EEEEEEEEEE  RR      RR  MM      MM  RR      RR      TT          ....


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  CTERMRT - CTERM remote terminal protocol module
0000     2          .IDENT  'V04-000'
0000     3
0000     4  $DEBUGDEF
0000     5
0000     6  ;
0000     7  ;**********************************************************************************
0000     8  ;*                                                                                *
0000     9  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                       *
0000    10  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                         *
0000    11  ;*  ALL RIGHTS RESERVED.                                                          *
0000    12  ;*                                                                                *
0000    13  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED          *
0000    14  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE        *
0000    15  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER        *
0000    16  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY        *
0000    17  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY        *
0000    18  ;*  TRANSFERRED.                                                                  *
0000    19  ;*                                                                                *
0000    20  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE         *
0000    21  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT        *
0000    22  ;*  CORPORATION.                                                                  *
0000    23  ;*                                                                                *
0000    24  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS        *
0000    25  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                       *
0000    26  ;*                                                                                *
0000    27  ;*                                                                                *
0000    28  ;**********************************************************************************
0000    29  ;
0000    30
0000    31  ;++
0000    32  ;
0000    33  ; FACILITY:
0000    34  ;
0000    35  ;       RTPAD in CLIUTL
0000    36  ;
0000    37  ; ABSTRACT:
0000    38  ;
0000    39  ;       This module contains all of the CTERM specific protocol needed
0000    40  ;       for running remote terminals.
0000    41  ;
0000    42  ; ENVIRONMENT:
0000    43  ;
0000    44  ;
0000    45  ;--
0000    46  ;
0000    47  ; AUTHOR: Jake VanNoy, CREATION DATE: 9-Sep-1982
0000    48  ;
0000    49  ; MODIFIED BY:
0000    50  ;
0000    51  ;       V03-008 JLV0389         Jake VanNoy             25-JUL-1984
0000    52  ;               Add read status code for SS$_OPINCOMPL.
0000    53  ;
0000    54  ;       V03-007 JLV0350         Jake VanNoy             10-APR-1984
0000    55  ;               Add setting of moredata flag in READ_DATA if not VAX host.
0000    56  ;               Add code to give proper message for known UNBINDs.
0000    57  ;
```

```
                     58 ;       V03-006 JLV0332         Jake VanNoy             28-FEB-1984
                     59 ;               Fix check that sets IO$M_BREAKTHRU if read is active.
                     60 ;               Change protocol in READ_DATA that returns cursor offset.
                     61 ;               Honor discard output in out of band path.
                     62 ;
                     63 ;       V03-005 JLV0322         Jake VanNoy             10-JAN-1984
                     64 ;               Fix bug in main QIO error path.  A privilege error
                     65 ;               on a SET MODE QIO would hang the remote process
                     66 ;               because the IOSB returned was 0.  Fixed by always
                     67 ;               placing QIO status in AST IOSB block.
                     68 ;               Fix bug in SET HOST/LOG that corrupts input logged.
                     69 ;               Remove local macros definitions.
                     70 ;
                     71 ;       V03-004 JLV0314         Jake VanNoy              7-DEC-1983
                     72 ;               Change handling of ^C normal echo. Use OOB_ECHO to
                     73 ;               flag whether one has been requested, clear this
                     74 ;               bit when ast is delivered.  Fix bug in input logging.
                     75 ;
                     76 ;       V03-003 JLV0292         Jake VanNoy             28-JUL-1983
                     77 ;               Add INIT message parsing. Remove input state on startup.
                     78 ;               Add message parameter 4 (VMS private) to INIT, this parameter
                     79 ;               will be used to init characteristics. This cuts the startup
                     80 ;               messages from 3 to 1. Add FLG$_x to $RTPADDEF.
                     81 ;               Add Set host/log code. Fix up out of band handling.
                     82 ;
                     83 ;       V03-002 JLV0264         Jake VanNoy             26-MAY-1983
                     84 ;               Add VMS tag to bind accept. Fix write flag processing.
                     85 ;
                     86 ;       V03-001 JLV0237         Jake VanNoy             29-MAR-1983
                     87 ;               Miscellanous bug fixes.
                     88 ;
                     89 ;**
                     90
                     91
                     92          .SBTTL  DECLARATIONS
                     93 ;
                     94 ; INCLUDE FILES:
                     95 ;
                     96
                     97          $RTPADDEF                   ; offsets and misc defintions
                     98          $TSADEF
                     99          .ENABLE SUPPRESSION
                    100          $IODEF
                    101          $MSGDEF                     ; mailbox message definitions
                    102          $SSDEF                      ; system status codes
                    103          $TRMDEF                     ; item list codes
                    104          $TTYDEFS                    ; Driver symbols
                    105
00000026            106 AST$T_BUF = CTP$B_PRO_MSGTYPE       ; ***
                    107
                    108 ;
                    109 ; MACROS:
                    110 ;
                    111
                    112 ;
                    113 ; EQUATED SYMBOLS:
                    114 ;
```

```
                     0000   115
                     0000   116 ; Parameters:
                     0000   117
00000001             0000   118         MIN_QUEUED_BUFS = 1             ; Controls buffering
00000003             0000   119         MAX_QUEUED_BUFS = 3             ; Controls buffering
                     0000   120
                     0000   121         ASSUME FLG$V_CTERM EQ 0         ; So BLB(S/C) can be done
                     0000   122 ;
                     0000   123 ; OWN STORAGE:
                     0000   124 ;
                     0000   125
00000000             0000   126 .PSECT          _RTPAD,LONG                     ; Read/Write data
                     0000   127
                     0000   128 ;
                     0000   129 ; *** MESSAGES DEFINED HERE MUST BE READ-ONLY ???
                     0000   130 ; Or only sent once...
                     0000   131 ;
                     0000   132
                     0000   133 ;
                     0000   134 ; Define INIT message
                     0000   135 ;
                     0000   136 CT_INIT_MSG:
00000026             0000   137                 .BLKB   CTP$B_PRO_MSGTYPE       ; AST header
0009                 0026   138 1$:              .WORD   PRO$C_DATA              ; protocol type, no flags
001B'                0028   139                 .WORD   10$-5$                  ; message length
0001                 002A   140 5$:              .WORD   CTP$C_MT_INIT           ; INIT message, no flags
01                   002C   141                 .BYTE   1                       ; version 1
00                   002D   142                 .BYTE   0                       ; Eco
00                   002E   143                 .BYTE   0                       ; Customer mod
0007                 002F   144                 .WORD   7                       ; VMS in revision field
00000037             0031   145                 .BLKB   6                       ; remainder of revision
                     0037   146
01                   0037   147                 .BYTE   1                       ; first parameter
02                   0038   148                 .BYTE   2                       ; Length
03F2                 0039   149                 .WORD   maxmsg-CTP$W_MSGSIZE    ; maximum "write to me" size?
                     003B   150
02                   003B   151                 .BYTE   2                       ; Second parameter
02                   003C   152                 .BYTE   2                       ; Length
03C0                 003D   153                 .WORD   maxmsg-CTP$T_SR_TERM-32 ; legal read size *** ?
                     003F   154
03                   003F   155                 .BYTE   3                       ; Third parameter
04                   0040   156                 .BYTE   4                       ; Length
0003FFFE             0041   157                 .LONG   ^B01111111111111111110  ; messages 1-17
                     0045   158 10$:
0000001F             0045   159 CT_INIT_MSGLEN = .-1$
                     0045   160
04                   0045   161 INIT_CHAR_STRT: .BYTE    4                   ; Fourth parameter (VMS ONLY!)
0C                   0046   162                 .BYTE   12                      ; Length
                     0047   163 CT_INIT_CHAR:
00000053             0047   164                 .BLKB   12                      ; characteristics goes here
                     0053   165
0000000E             0053   166 CT_INIT_CHAR_MSGLEN =.-INIT_CHAR_STRT
                     0053   167
                     0053   168 ;
                     0053   169 ; Define a message that says "Unsolicited Input"
                     0053   170 ;
                     0053   171 CT_UNSDAT_MSG::                             ; Set up unsolicited data message
```

```
00000079  0053   172                    .BLKB   CTP$B_PRO_MSGTYPE       ; Header
    0009  0079   173                    .WORD   PRO$C_DATA              ; data message
    0002' 007B   174                    .WORD   10$-5$                  ; Message length
      0E  007D   175 5$:                .BYTE   CTP$C_MT_INP_STATE      ; Input state
      01  007E   176                    .BYTE   CTP$M_IS_NONZERO        ; Change from zero to non-zero
          007F   177 10$:
          007F   178
          007F   179 CT_BIND_ACC_MSG::
000000A5  007F   180                    .BLKB   CTP$B_PRO_MSGTYPE       ; Header
      04  00A5   181 1$:                .BYTE   PRO$C_ACCEPT            ; Bind accept message type
      02  00A6   182                    .BYTE   2                       ; version number
   00 03  00A7   183                    .BYTE   3,0                     ; ECO, customer ECO ***
    0007  00A9   184                    .WORD   7                       ; hi, we're VMS!
000000B3  00AB   185                    .BLKB   8                       ; revision
    0000  00B3   186                    .WORD   0                       ; ID
      00  00B5   187                    .BYTE   0                       ; Options
00000011  00B6   188 CT_BIND_MSGLEN == .-1$
          00B6   189
          00B6   190 UNBIND_MSG:
000000DC  00B6   191                    .BLKB   CTP$B_PRO_MSGTYPE       ; Header ???
      02  00DC   192                    .BYTE   PRO$C_UNBIND            ; message type
    0000  00DD   193 UNBIND_WHY:        .WORD   0                       ; reason code ** must be filled in *
00000029  00DF   194 UNBIND_MSGLEN = .-UNBIND_MSG
          00DF   195
          00DF   196 TERMSET:
00000020  00DF   197             .LONG   32                             ; 32 bytes of data
000000E7' 00E3   198             .ADDRESS 10$                           ; Address of data
00000107  00E7   199 10$:        .BLKB   32                             ;
          0107   200
          0107   201
          0107   202 OUTBAND_SET:
00000000 00000000  0107   203             .LONG   0,0                   ; enabled characters
00000000 00000000  010F   204             .LONG   0,0                   ; includes
00000000 00000000  0117   205             .LONG   0,0                   ; aborts
00000000 00000000  011F   206             .LONG   0,0                   ; discard and echo
          0127   207
          0127   208 ASSUME OOB_LEN EQ 8*4   ; assume length of storage
          0127   209
          0127   210 OUTBAND_NEW::
00000000 00000000  0127   211             .LONG   0,0                   ; enabled characters
00000000 00000000  012F   212             .LONG   0,0                   ; includes
00000000 00000000  0137   213             .LONG   0,0                   ; aborts
00000000 00000000  013F   214             .LONG   0,0                   ; discard
          0147   215
          0147   216 RD_STAT_TBL:                                       ; no particular order necessary
          0147   217
     0000 0001   0147   218             .WORD   SS$_NORMAL,     CTP$M_RD_NORMAL       ; Success (most likely)
     0005 022C   014B   219             .WORD   SS$_TIMEOUT,    CTP$M_RD_TIMEOUT      ; Timeout
     0003 0611   014F   220             .WORD   SS$_CONTROLY,   CTP$M_RD_OUTBAND      ; out of band
     0003 C651   0153   221             .WORD   SS$_CONTROLC,   CTP$M_RD_OUTBAND      ; out of band *** (^Y on ctd
     0006 692C   0157   222             .WORD   SS$_ABORT,      CTP$M_RD_UNREAD       ; $CANCEL on channel
     0002 003C   015B   223             .WORD   SS$_BADESCAPE,  CTP$M_RD_INVESC       ; invalid escape
     0008 01FC   015F   224             .WORD   SS$_PARTESCAPE, CTP$M_RD_ABSTOKEN     ; partial *** semantics???
     000C 01F4   0163   225             .WORD   SS$_PARITY,     CTP$M_RD_PARITY       ; parity error
     000D 0838   0167   226             .WORD   SS$_DATAOVERUN, CTP$M_RD_OVERUN       ; receiver over run
     000E 02D4   016B   227             .WORD   SS$_OPINCOMPL,  14                    ; ** need CTP code here
          016F   228
```

```
                     016F   229 ; *** PARITY ERRORS NOT DONE HERE YET ...
                     016F   230
                     016F   231 RD_STAT_END:
                0000 016F   232         .WORD   CTP$M_RD_NORMAL         ; Catch all
                     0171   233
                     0171   234 READNETFLAG:
                  00 0171   235         .BYTE   0                       ; read from net flag
                     0172   236
                     0172   237 QUEUED_BUFS:
            00000000 0172   238         .LONG   0
                     0176   239
                     0176   240 FLUSH_STATUS:
            00000000 0176   241         .LONG   0
                     017A   242
                     017A   243 SAVE_WR_POSTFIX:
                  00 017A   244         .BYTE   0                       ; Save postfix from first write
                     017B   245
                     017B   246 SAVE_WR_FLAGS:
                0000 017B   247         .WORD   0                       ; Save write flags
                     017D   248
                     017D   249 CTERM_VERSION:
                  00 017D   250         .BYTE   0
                     017E   251 CTERM_ECO:
                  00 017E   252         .BYTE   0
                     017F   253
                     017F   254 ;
                     017F   255 ; Data from INIT Message
                     017F   256 ;
                     017F   257
                     017F   258 MAXSENDMSG:
            00000000 017F   259         .LONG   0
                     0183   260 MAXREAD:
            00000000 0183   261         .LONG   0
                     0187   262 LEGALMSG:
            000001A7 0187   263         .BLKB   32                      ; This is maximum
                     01A7   264 ;
                     01A7   265 ; LOGGING OWN STORAGE
                     01A7   266 ;
            000005C1 01A7   267 LOG_BUF:        .blkb   maxmsg
            00000000 05C1   268 LOG_ADDR:       .long   0
                     05C5   269
                     05C5   270 .if df debug
                     05C5   271 neton:           .ascic  /Turn net on/
                     05C5   272 netoff:          .ascic  /Turn net off/
                     05C5   273 .endc
                     05C5   274
                00000000    275         .PSECT  RTPAD,NOWRT             ; Code
```

```
                     0000   277          .SBTTL  CTERM_LINKRECV - Cterm message on link received
                     0000   278
                     0000   279 ;++
                     0000   280 ;
                     0000   281 ; FUNCTIONAL DESCRIPTION:
                     0000   282 ;
                     0000   283 ; Decide on message type whether to queue packet or process immediately.
                     0000   284 ;
                     0000   285 ; CALLING SEQUENCE:
                     0000   286 ;       BSBW    CTERM_LINKRECV
                     0000   287 ;
                     0000   288 ; INPUT PARAMETERS:
                     0000   289 ;
                     0000   290 ;       R0 - AST control block
                     0000   291 ;
                     0000   292 ; IMPLICIT INPUTS:
                     0000   293 ;
                     0000   294 ;       WRITEQIO queue
                     0000   295 ;       READQIO queue
                     0000   296 ;
                     0000   297 ; OUTPUT PARAMETERS:
                     0000   298 ;
                     0000   299 ;       NONE
                     0000   300 ;
                     0000   301 ; IMPLICIT OUTPUTS:
                     0000   302 ;
                     0000   303 ; An entry may be added to a queue.
                     0000   304 ;
                     0000   305 ; COMPLETION CODES:
                     0000   306 ;
                     0000   307 ; SIDE EFFECTS:
                     0000   308 ;
                     0000   309 ;--
                     0000   310
                     0000   311 NOT_DATA:
          02   91    0000   312          CMPB    #PRO$C_UNBIND,-
        26 A0         0002   313                  CTP$B_PRO_MSGTYPE(R0)   ; unbind message?
          37   13     0004   314          BEQL    20$                     ; If yes, die
00000000'8F    DD     0006   315          PUSHL   #REM$_UNKMSG            ; Unknown message
00000000'GF  01    FB 000C   316          CALLS   #1,G^LIB$SIGNAL         ; Signal
                     0013   317          QUIT    #SS$_LINKDISCON         ; quit
               04    003C   318          RET
                     003D   319 20$:
        0B5D   30    003D   320          BSBW    UNBIND_RECEIVED         ; unbind received
               04    0040   321          RET
                     0041   322
                     0041   323 CTERM_LINKRECV::
                     0041   324
                     0041   325 ;+
                     0041   326 ;
                     0041   327 ; Valid message types here are:
                     0041   328 ;
                     0041   329 ; ctp$c_mt_init          (1)     Initiate            (H <---> S)
                     0041   330 ; ctp$c_mt_start_rd      (2)     Start Read          (H  ---> S)
                     0041   331 ; ctp$c_mt_unread        (5)     Unread              (H  ---> S)
                     0041   332 ; ctp$c_mt_clr_input     (6)     Clear Input         (H  ---> S)
                     0041   333 ; ctp$c_mt_write         (7)     Write               (H  ---> S)
```

```
                                0041   334 ; ctp$c_mt_read_char    (10)     Read Characteristics   (H  ---> S)
                                0041   335 ; ctp$c_mt_char         (11)     Characteristics        (H <---> S)
                                0041   336 ; ctp$c_mt_check_inp    (12)     Check Input            (H  ---> S)
                                0041   337 ;
                                0041   338 ;-
                     0E A0  B4  0041   339         CLRW    AST$W_OFFSET(R0)          ; Clear initial offset
           06 A0     02  A3  0044   340         SUBW3   #2,AST$Q_IOSB+2(R0),-     ; Compute size of message minus ***
                  10 A0        0048   341                 AST$W_BUFSIZ(R0)          ;  foundation overhead,
                               004A   342
                               004A   343 ;
                               004A   344 ; R0 IS ALWAYS USED TO CHECK PROTOCOL MESSAGE TYPE (FOUNDATION BYTE)
                               004A   345 ; THIS IS BECAUSE THERE ONLY ONE OF THESE PER MESSAGE
                               004A   346 ;
                               004A   347
                               004A   348 ; check pro_msgtypes here (SHOULD ALWAYS BE CTERM HERE, UNLESS DISCONNECT)
                               004A   349
                        09  91   004A   350         CMPB    #PRO$C_DATA,-
                     26 A0        004C   351                 CTP$B_PRO_MSGTYPE(R0)     ; data (cterm) message?
                        B0  12   004E   352         BNEQ    NOT_DATA                  ; If yes, continue
                               0050   353
                               0050   354         CASE    CTP$B_MSGTYPE(R0),-       ; Case on message type
                               0050   355                 <10$,-                    ; 0     invalid
                               0050   356                 INIT_RCV,-                ; 1     init
                               0050   357                 20$,-                     ; 2     start_read
                               0050   358                 10$,10$,-                 ; 3,4   read_data,out_of_band
                               0050   359                 UNREAD_RCV,-              ; 5     unread request
                               0050   360                 CLR_INPUT_RCV,-           ; 6     clear input request
                               0050   361                 100$,-                    ; 7     write
                               0050   362                 10$,10$,-                 ; 8,9   write complete,discard output
                               0050   363                 20$,-                     ; 10    read_char
                               0050   364                 20$,-                     ; 11    set char
                               0050   365                 20$,-                     ; 12    check input
                               0050   366                 10$,10$,-                 ; 13,14 input count, input state
                               0050   367                 100$,-                    ; 15    vms qio
                               0050   368                 10,-                      ; 16    vms brdcst (invalid here)
                               0050   369                 20$-                      ; 17    vms read verify
                               0050   370                 >,-
                               0050   371                 TYPE = B                  ; Byte field
                               0079   372 ;
                               0079   373 ; Ignore invalid message
                               0079   374 ;
                               0079   375
                               0079   376 10$:
                        007A   31   0079   377         BRW     ERROR_RCV                 ; Signal message
                               05   007C   378         RSB                               ; Just return
                               007D   379 ;
                               007D   380 ; READ, SET, SENSE
                               007D   381 ;
                               007D   382 20$:                                      ; Read or Set/Sense mode request
                00000000'EF    D5   007D   383         TSTL    READQIO                   ; Read outstanding?
                          17   12   0083   384         BNEQ    30$                       ; Branch if yes
                00000000'EF    D5   0085   385         TSTL    WRITEQIO                  ; Write outstanding?
                          0F   12   008B   386         BNEQ    30$                       ; Branch if yes
                00000000'EF    D6   008D   387         INCL    READQIO                   ; Prevent others from starting up
           00000112'EF    00   FB   0093   388         CALLS   #0,CTERM_PROCMSG          ; Process now
                          33   11   009A   389         BRB     CTERM_LINKRECV_EXIT       ; Exit
                               009C   390 30$:
```

```
                00000172'EF   D6  009C   391          INCL    QUEUED_BUFS                ; Add to count now in queue
00000004'FF       60    0E   00A2   392          INSQUE  (R0),@READQ+4               ; Stick at end of queue
                  24    11   00A9   393          BRB     CTERM_LINKRECV_EXIT        ; Exit
                           00AB   394 ;
                           00AB   395 ; WRITE
                           00AB   396 ;
                           00AB   397 100$:
                00000000'EF   D5  00AB   398          TSTL    WRITEQIO                   ; Write outstanding?
                           0F   12  00B1   399          BNEQ    110$                       ; Branch if yes
                00000000'EF   D6  00B3   400          INCL    WRITEQIO                   ; Prevent others from starting up
00000112'EF       00    FB   00B9   401          CALLS   #0,CTERM_PROCMSG           ; Process now
                  0D    11   00C0   402          BRB     CTERM_LINKRECV_EXIT        ; Exit
                           00C2   403 110$:
                00000172'EF   D6  00C2   404          INCL    QUEUED_BUFS                ; Add to count now in queue
00000004'FF       60    0E   00C8   405          INSQUE  (R0),@WRITEQ+4             ; Stick at end of queue
                           00CF   406
                           00CF   407 CTERM_LINKRECV_EXIT:
                           00CF   408
                  03    D1   00CF   409          CMPL    #MAX_QUEUED_BUFS,-
                00000172'EF   00D1   410                  QUEUED_BUFS                ; Check amount queued
                  01    15   00D6   411          BLEQ    10$                        ; Branch if lots queued
                        05   00D8   412          RSB                                ; not real busy, return and do
                           00D9   413                                             ;  another read from the net
                           00D9   414 ;
                           00D9   415 ; QIO's to terminal are getting behind data from net, don't return
                           00D9   416 ; to LINKRECV in RTPAD so that another read from the net will occur.
                           00D9   417 ; Instead, set a flag that shows reading from the net must be scheduled.
                           00D9   418 ;
                           00D9   419 10$:
                00000171'EF   96  00D9   420          INCB    READNETFLAG                ; Set flag
                           00DF   421
                           00DF   422 .IF DF DEBUG
                           00DF   423          movab   netoff,r0          ; ** Debug
                           00DF   424          bsbw    log_ascic          ; *** Debug
                           00DF   425 .ENDC
                           00DF   426
                        04   00DF   427          RET                                ; Exit from AST
                           00E0   428
                           00E0   429 INIT_RCV:
                           00E0   430 CLR_INPUT_RCV:
                           00E0   431 UNREAD_RCV:
                           00E0   432
                00000000'EF   D6  00E0   433          INCL    READQIO            ; *** yuck
                           00E6   434
60              00000000'EF   9E  00E6   435          MOVAB   QIODONE,AST$L_STATE(R0) ; Set next state after this
00000112'EF       00    FB   00ED   436          CALLS   #0,CTERM_PROCMSG           ; Process NOW
                  D9    11   00F4   437          BRB     CTERM_LINKRECV_EXIT        ; Exit
                           00F6   438
                           00F6   439 ERROR_RCV:
                     5B    50   D0   00F6   440          MOVL    R0,R11
50              00000000'8F   D0   00F9   441          MOVL    #REM$_BADMSG,R0            ; Bad message
                           0100   442
                           0100   443 BADFIELD_ERROR:
                           0100   444
                        50   DD   0100   445          PUSHL   R0                         ; reason code
                        7E   D4   0102   446          CLRL    -(SP)                      ; no fao
                00000000'8F   DD   0104   447          PUSHL   #REM$_PROTERR              ; protocol error
```

```
                00000000'GF     03   FB  010A    448          CALLS   #3,G^LIB$SIGNAL
                                     05  0111    449          RSB
                                         0112    450
```

```
                     0112   452          .SBTTL  CTERM_PROCMSG - Process message recieved from link
                     0112   453
                     0112   454 ;++
                     0112   455 ;
                     0112   456 ; FUNCTIONAL DESCRIPTION:
                     0112   457 ;
                     0112   458 ; Process message recieved from link. Take a protocol packet received
                     0112   459 ; from the net and translate that into a terminal action on this end.
                     0112   460 ;
                     0112   461 ; CALLING SEQUENCE:
                     0112   462 ;       CALLS   #0,CTERM_PROCMSG
                     0112   463 ;
                     0112   464 ; INPUT PARAMETERS:
                     0112   465 ;
                     0112   466 ;       R0 points to AST control block.
                     0112   467 ;
                     0112   468 ; IMPLICIT INPUTS:
                     0112   469 ;       NONE
                     0112   470 ;
                     0112   471 ; OUTPUT PARAMETERS:
                     0112   472 ;       NONE
                     0112   473 ;
                     0112   474 ; IMPLICIT OUTPUTS:              ???
                     0112   475 ;       READQIO
                     0112   476 ;       WRITEQIO
                     0112   477 ;       RETSTATUS
                     0112   478 ;
                     0112   479 ; COMPLETION CODES:
                     0112   480 ;       NONE
                     0112   481 ;
                     0112   482 ; SIDE EFFECTS:
                     0112   483 ;
                     0112   484 ;       A QIO to the terminal may be performed.
                     0112   485 ;
                     0112   486 ;--
                     0112   487
                     0112   488 CTERM_PROCMSG:
                OFFC 0112   489          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Register save mask
                     0114   490 ;+
                     0114   491 ;
                     0114   492 ; Valid message types here are:
                     0114   493 ;
                     0114   494 ; ctp$c_mt_init          (1)     Initiate                (H <---> S)
                     0114   495 ; ctp$c_mt_start_rd      (2)     Start Read              (H  ---> S)
                     0114   496 ; ctp$c_mt_unread        (5)     Unread                  (H  ---> S)
                     0114   497 ; ctp$c_mt_clr_input     (6)     Clear Input             (H  ---> S)
                     0114   498 ; ctp$c_mt_write         (7)     Write                   (H  ---> S)
                     0114   499 ; ctp$c_mt_read_char     (10)    Read Characteristics    (H  ---> S)
                     0114   500 ; ctp$c_mt_char          (11)    Characteristics         (H <---> S)
                     0114   501 ; ctp$c_mt_check_inp     (12)    Check Input             (H  ---> S)
                     0114   502 ;
                     0114   503 ;-
                     0114   504
5B   0E A0   3C      0114   505          MOVZWL  AST$W_OFFSET(R0),R11    ; Offset to current message
   5B   50   C0      0118   506          ADDL2   R0,R11                  ; Add in base
                     011B   507
        51   7C      011B   508          CLRQ    R1                      ; Clear R1,R2
```

```
                 53   7C  011D   509          CLRQ    R3                          ; Clear R3,R4
                 55   7C  011F   510          CLRQ    R5                          ; Clear R5,R6
     57  00000000'EF   3C  0121   511          MOVZWL  READCHAN,R7                 ; Assume read channel
     60  00000000'EF   9E  0128   512          MOVAB   QIODONE,AST$L_STATE(R0)     ; Assume next state after this
                       012F   513
                       012F   514          CASE    CTP$B_MSGTYPE(R11),-        ; Case on message type
                       012F   515                  <10$,-                      ; 0 invalid
                       012F   516                  INIT_MSG,-                  ; 1
                       012F   517                  START_READ_MSG,-            ; 2
                       012F   518                  10$,10$,-                   ; 3,4
                       012F   519                  UNREAD_MSG,-                ; 5
                       012F   520                  CLR_INPUT_MSG,-             ; 6
                       012F   521                  WRITE_MSG,-                 ; 7
                       012F   522                  10$,10$,-                   ; 8,9
                       012F   523                  READ_CHAR_MSG,-             ; 10
                       012F   524                  CHAR_MSG,-                  ; 11
                       012F   525                  CHECK_INP_MSG,-             ; 12
                       012F   526                  10$,10$,-                   ; 13,14
                       012F   527                  VMSQIO_MSG,-                ; 15
                       012F   528                  10$,START_READ_MSG-         ; 16,17 (read verify)
                       012F   529                  >,-
                       012F   530                  TYPE = B                    ; Byte field
                       0158   531 ;
                       0158   532 ; Ignore invalid message
                       0158   533 ;
                       0158   534
                       0158   535 10$:
                    04 0158   536          RET                                 ; Just return
                       0159   537
                       0159   538 VMSQIO_MSG:
                       0159   539
     00000000'EF   D6  0159   540          INCL    WRITEQIO                    ; *** should this be here?
            FE9E'  30  015F   541          BSBW    CTERM_VMSQIO                ; handle qio
             0500  31  0162   542          BRW     CTERM_PROCMSG_QIO           ; do it
                       0165   543
```

```
                     0165   545 INIT_MSG:                                   ; message type = 1
                     0165   546
                  50 DD  0165   547         PUSHL   R0                      ; Save
                2C AB B0 0167   548         MOVW    CTP$B_IN_VERSION(R11),-
             017D'CF     016A   549                 W^CTERM_VERSION          ; Version and ECO
                     016D   550         ;
                     016D   551         ; Parse INIT message
                     016D   552         ;
          56    37 AB 9E 016D   553         MOVAB   CTP$B_IN_PARMTYPE(R11),R6 ; address of first parameter
          57    2A AB 9E 0171   554         MOVAB   CTP$B_MSGTYPE(R11),R7   ; address of base of message
          5A    28 AB 3C 0175   555         MOVZWL  CTP$W_MSGSIZE(R11),R10  ; size of message
             5A    57 C0 0179   556         ADDL    R7,R10                  ; end of message
                     017C   557
             56    5A D1 017C   558 10$:     CMPL    R10,R6                  ; compare to end
                   52 15 017F   559         BLEQ    90$                     ; exit if done
          57    01 A6 9A 0181   560         MOVZBL  1(R6),R7                ; Size of parameter
             59    02 D0 0185   561         MOVL    #2,R9                   ; Default maximum storage area size
                     0188   562         CASE    (R6),-
                     0188   563                 <20$,30$,40$>,-             ;
                     0188   564                 TYPE=B,LIMIT=#1             ; Limit is really base?
                   E8 11 0192   565         BRB     10$                     ; error here...
                     0194   566
       58 0000017F'EF 9E 0194   567 20$:     MOVAB   MAXSENDMSG,R8
                   13 11 019B   568         BRB     70$
                     019D   569
       58 00000183'EF 9E 019D   570 30$:     MOVAB   MAXREAD,R8              ; Shouldn't really get this in server
                   0A 11 01A4   571         BRB     70$
                     01A6   572
       58 00000187'EF 9E 01A6   573 40$:     MOVAB   LEGALMSG,R8             ; Legal messages
             59    08 D0 01AD   574         MOVL    #8,R9
                     01B0   575
          56    02 A6 9E 01B0   576 70$:     MOVAB   2(R6),R6                ; bias
             59    57 D1 01B4   577         CMPL    R7,R9                   ; check sizes
                   0C 1B 01B7   578         BLEQU   76$                     ; branch if it fits
                     01B9   579         ; minor error here, about to lose data
          7E 57    59 C3 01B9   580         SUBL3   R9,R7,-(SP)             ; calculate remainder
             57    59 D0 01BD   581         MOVL    R9,R7                   ; set size
                   59 8ED0 01C0   582         POPL    R9                      ; remainder
                   02 11 01C3   583         BRB     80$                     ;
                     01C5   584
                   59 D4 01C5   585 76$:     CLRL    R9
                     01C7   586
             88    86 90 01C7   587 80$:     MOVB    (R6)+,(R8)+             ; set up
                FA 57 F5 01CA   588         SOBGTR  R7,80$                  ;
                     01CD   589
             56  6649 9E 01CD   590         MOVAB   (R6)[R9],R6             ; add remainder
                   A9 11 01D1   591         BRB     10$
                     01D3   592 90$:
                     01D3   593         ;
                     01D3   594         ; Send INIT message to HOST now - receiving this message
                     01D3   595         ; will start up LOGIN on VMS.
                     01D3   596         ;
       50 00000000'EF 9E 01D3   597         MOVAB   CT_INIT_MSG,R0          ; AST block address
             51    1F 9A 01DA   598         MOVZBL  #CT_INIT_MSGLEN,R1      ; Length
          52    26 A0 9E 01DD   599         MOVAB   CTP$B_PRO_MSGTYPE(R0),R2 ; message address
                     01E1   600         ;
                     01E1   601         ; Send CHAR message to init characteristics (VMS TO VMS ONLY)
```

```
                                01E1    602 ;
                          04    E1  01E1    603         BBC     #FLG$V_VAXHOST,-
                    20 0000'CF        01E3    604                 W^CTERM_FLAGS0$         ; Branch if not VAX to VAX
                53  00000047'EF    9E  01E7    605         MOVAB   CT_INIT_CHAR,R3         ; block address
                63  00000000'EF    7D  01EE    606         MOVQ    CHAR_BLOCK,(R3)         ; first 8 bytes
            08 A3   00000008'EF    D0  01F5    607         MOVL    CHAR_BLOCK+8,8(R3)      ; second 4 bytes
                        54      0E  9A  01FD    608         MOVZBL  #CT_INIT_CHAR_MSGLEN,R4 ; Length
                        51      54  C0  0200    609         ADDL    R4,R1                   ; Add to length
                    28 A0       54  A0  0203    610         ADDW    R4,CTP$W_MSGSIZE(R0)    ; Add to length
                                        0207    611 100$:
                            FDF6'   30  0207    612         BSBW    WRITE_TO_NET_SYNC       ; Write to net synchronously
                                        020A    613
                              50 8ED0   020A    614         POPL    R0
                            04CF    31  020D    615         BRW     CTERM_PROCMSG_EXIT      ; Exit back to common code
                                        0210    616
```

```
                                0210   618 START_READ_MSG:                         ; message type = 2
                                0210   619
          00000000'EF    D6     0210   620         INCL    READQIO                 ; Set "read active"
                                0216   621
                         AA     0216   622         BICW    #<FLG$M_CTRL_O!-
                                0217   623                 FLG$M_CTRL_CY>,-
00000000'EF    06               0217   624                 CTERM_FLAG              ; Clear ^O and flushing
                                021D   625
            16 A0      D4      021D   626         CLRL    AST$L_ITMLST(R0)        ; clear itemlist pointer
         58    37      9A      0220   627         MOVZBL  #IO$_READPROMPT,R8      ; Set function code
         18    00      EF      0223   628         EXTZV   #0,#8+16,-
       51    2B AB             0226   629                 CTP$L_SR_FLAGS(R11),R1  ; Flags (3 bytes)
                     08  EF     0229   630         EXTZV   #CTP$V_SR_CONTROL,-
                     03         022B   631                 #CTP$S_SR_CONTROL,-
            5A       51        022C   632                 R1,R10                  ; Extract control character flags
                     16  13     022E   633         BEQL    30$                     ; If none, branch
                                0230   634 ;
                                0230   635 ; Decide on function code. (note: READPROMPT and TTYREADPALL may be enough)????
                                0230   636 ;
                                0230   637
                                0230   638 ; *** case on type R10?
                                0230   639
            5A       03  91     0230   640         CMPB    #CTP$C_SR_ALLBUTX,R10   ; read passall (sort of)?
                     05  12     0233   641         BNEQ    20$                     ; no, branch
            58       3B  90     0235   642         MOVB    #IO$_TTYREADPALL,R8     ; Set function code
                     0C  11     0238   643         BRB     30$
                                023A   644 20$:
            5A       02  91     023A   645         CMPB    #CTP$C_SR_EDIT,R10      ; Read edit characters?
                     07  12     023D   646         BNEQ    25$                     ; If not, signal
         58    0200 8F  A8     023F   647         BISW    #IO$M_NOFILTR,R8        ; Set read edit characters
                     00  11     0244   648         BRB     30$
                                0246   649 25$:
                                0246   650         ; *** whats left?
                                0246   651 ;
                                0246   652 ; Map flags into function code modifiers
                                0246   653 ;
                                0246   654 30$:
      05 51      0B  E1         0246   655         BBC     #CTP$V_SR_NOECHO,R1,40$ ; No echo?
   58    0040 8F  A8             024A   656         BISW    #IO$M_NOECHO,R8         ;
                                024F   657 40$:
      09 51      0D  E1         024F   658         BBC     #CTP$V_SR_TIMED,R1,60$  ; Timed?
   58    0080 8F  A8             0253   659         BISW    #IO$M_TIMED,R8          ;
     53    32 AB  B0             0258   660         MOVW    CTP$W_SR_TIMEOUT(R11),R3        ; Set timeout value
                                025C   661 60$:
      05 51      02  E1         025C   662         BBC     #CTP$V_SR_PURGE,R1,80$  ; Purge typeahead?
   58    0800 8F  A8             0260   663         BISW    #IO$M_PURGE,R8          ;
                                0265   664 80$:
                     0C  E0     0265   665         BBS     #CTP$V_SR_TRMECHO,-
            05 51               0267   666                 R1,100$                 ; Terminator echo?
   58    1000 8F  A8             0269   667         BISW    #IO$M_TRMNOECHO,R8      ; no
                                026E   668 100$:
                     07  E1     026E   669         BBC     #CTP$V_SR_CVTLOW+1,-    ; hack assumes values of CVTLOW field
            05 51               0270   670                 R1,120$                 ; convert lower to upper?
   58    0100 8F  A8             0272   671         BISW    #IO$M_CVTLOW,R8         ;
                                0277   672 120$:
                     11  E1     0277   673         BBC     #CTP$V_SR_ESCAPE,-      ; Escape?
            05 51               0279   674                 R1,140$                 ;
```

```
                58   4000 8F    A8  027B   675          BISW    #IO$M_ESCAPE,R8         ;  Set read w/escape
                                    0280   676 140$:
                          12    EF  0280   677          EXTZV   #CTP$V_SR_NOEDIT,-
                7E    51    02        0282   678                  #2,R1,-(SP)             ; Extract VAX specific bits
                      0F    8E  F0    0285   679          INSV    (SP)+,#TRM$V_TM_NOEDIT,-
                      58    02        0288   680                  #2,R8                   ; And put them where they belong
                                      028A   681 160$:
                                      028A   682 ;
                                      028A   683 ; Set up terminator mask
                                      028A   684 ;
                                      028A   685
                  55    3A AB    9E  028A   686          MOVAB   CTP$T_SR_TERM(R11),R5   ; Fetch terminator field address
                            02    91  028E   687          CMPB    #CTP$C_MT_START_RD,-
                          2A AB       0290   688                  CTP$B_MSGTYPE(R11)      ; Normal read
                            0A    13  0292   689          BEQL    200$                    ; Branch if yes
                                      0294   690          ;
                                      0294   691          ; Read verify
                                      0294   692          ;
                  55    42 AB    9E  0294   693          MOVAB   CTP$T_SR2_TERM(R11),R5  ; Fetch terminator field address
                        3E AB    F0  0298   694          INSV    CTP$W_SR2_EDITFLAGS(R11),-
                58    02    11        029B   695                  #TRM$V_TM_R_JUST,#2,R8  ; Set extra read flags
                                      029E   696 200$:
                      56    85  9A  029E   697          MOVZBL  (R5)+,R6                ; Length of terminator mask
                            0E  EF  02A1   698          EXTZV   #CTP$V_SR_TERM_SET,-
                            02        02A3   699                  #CTP$S_SR_TERM_SET,-
                      5A    51        02A4   700                  R1,R10                  ; Extract terminator set flags
                                      02A6   701          ASSUME  CTP$C_SR_PREVTERM EQ 0  ; Assume 0 is previous terminator
                            1A    13  02A6   702          BEQL    210$                    ; Branch if not zero
                      5A    02    91  02A8   703          CMPB    #CTP$C_SR_NORMTERM,R10  ; Use this terminator set?
                            1A    13  02AB   704          BEQL    220$                    ; yes, continue with R4 as zero
                                      02AD   705
                      5A    01    91  02AD   706          CMPB    #CTP$C_SR_THISTERM,R10  ; Use this terminator set?
                            15    12  02B0   707          BNEQ    220$                    ; unknown value - use normal set
                            54  00DF'CF  9E  02B2   708          MOVAB   W^TERMSET,R4            ; Use this set
                                  3F  BB  02B7   709          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save
08 A4    20   00    65          56  2C  02B9   710          MOVC5   R6,(R5),#0,#32,8(R4)    ; Copy terminator mask
                                  3F  BA  02C0   711          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore
                                      02C2   712 210$:
                            54  00DF'CF  9E  02C2   713          MOVAB   W^TERMSET,R4            ; Use previous set (VMS won't send this)
                                      02C7   714 ;
                                      02C7   715 ; Set up prompt and buffer
                                      02C7   716 ;
                                      02C7   717 220$:
                            55    56  C0  02C7   718          ADDL    R6,R5                     ; Address of read data (prompt)
                              56    34 AB  3C  02CA   719          MOVZWL  CTP$W_SR_END_PRMT(R11),R6 ; Length of prompt
                            51    56  55  C1  02CE   720          ADDL3   R5,R6,R1                  ; Begin of buffer for read data
                                      02D2   721
                                  03  E1  02D2   722          BBC     #FLG$V_LOGGING,-
                                0F 0000'CF        02D4   723                  W^CTERM_FLAG,222$       ; Branch if not logging
                                  3F  BB  02D8   724          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; ** LOG prompt
                01A7'CF   65          56  28  02DA   725          MOVC3   R6,(R5),W^LOG_BUF       ; **
                      05C1'CF   53        D0  02E0   726          MOVL    R3,W^LOG_ADDR           ; **
                                  3F  BA  02E5   727          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; **
                                      02E7   728 222$:
```

```
                        02E7   730          ; At this point:
                        02E7   731          ;
                        02E7   732          ; R1 - address of data or initial string
                        02E7   733          ; R2 - scratch
                        02E7   734          ; R3 - timeout value
                        02E7   735          ; R4 - address of terminator set
                        02E7   736          ; R5 - Address of prompt string
                        02E7   737          ; R6 - length of prompt
                        02E7   738          ; R7 - readchan
                        02E7   739          ; R8 - function code
                        02E7   740          ; R9, R10 - scratch
                        02E7   741          ; R11 - CTP
                        02E7   742          ;in CTP:
                        02E7   743          ; SR_END_PRMT = length of prompt
                        02E7   744          ; SR_END_DATA = length of prompt + length of inioffset
                        02E7   745          ; SR_STR_DISP = 0 (normal)
                        02E7   746          ;               n inioffset (initial string non-zero implies item list)
                        02E7   747          ;
        59    36 AB   3C    02E7   748          MOVZWL  CTP$W_SR_STR_DISP(R11),R9  ; fetch INIOFFSET parameter
        5A    30 AB   3C    02EB   749          MOVZWL  CTP$W_SR_END_DATA(R11),R10 ; Fetch end of data length
           5A    56   D1    02EF   750          CMPL    R6,R10                     ; init string present?
                 12   12    02F2   751          BNEQ    230$                       ; Yes, do read itemlist
     58 7FFF8000 8F   D3    02F4   752          BITL    #^XFFFF8TRM$V_TM_NOEDIT,R8 ; high bits in R8 set?
                 09   12    02FB   753          BNEQ    230$                       ; other bits set, must use itemlist
                 11   91    02FD   754          CMPB    #CTP$C_MT_VMS_READVFY,-
                    2A AB      02FF   755                  CTP$B_MSGTYPE(R11)         ; see if read verify
                 03   13    0301   756          BEQL    230$                       ; must use itemlist if true
                  00C0   31    0303   757          BRW     300$                       ; nothing special, do normal read
                        0306   758 230$:
                        0306   759          ;
                        0306   760          ; Allocate itemlist
                        0306   761          ;
           50    DD    0306   762          PUSHL   R0                         ; Save R0
        FCF5'    30    0308   763          BSBW    GETBUF                     ; Get buffer
     5A    50    D0    030B   764          MOVL    R0,R10                     ; Address of itemlist
           50 8ED0    030E   765          POPL    R0                         ; Restore R0
  16 A0    5A    D0    0311   766          MOVL    R10,AST$L_ITMLST(R0)       ; Save address of itemlist
                        0315   767          ;
                        0315   768          ; TRM$_PROMPT
                        0315   769          ;
           5A    DD    0315   770          PUSHL   R10                        ; Save address
     8A    56    B0    0317   771          MOVW    R6,(R10)+                  ; prompt length
     8A    04    B0    031A   772          MOVW    #TRM$_PROMPT,(R10)+        ; prompt
     8A    55    D0    031D   773          MOVL    R5,(R10)+                  ; address of prompt
           8A    D4    0320   774          CLRL    (R10)+                     ; mbz
           55 8ED0    0322   775          POPL    R5                         ; address of itemlist
                        0325   776          ;
                        0325   777          ; TRM$_TIMEOUT
                        0325   778          ;
  0A 58    07    E1    0325   779          BBC     #IO$V_TIMED,R8,240$        ; Branch if timeout not requested
           8A    B4    0329   780          CLRW    (R10)+                     ; no value
     8A    02    B0    032B   781          MOVW    #TRM$_TIMEOUT,(R10)+       ; timeout
     8A    53    D0    032E   782          MOVL    R3,(R10)+                  ; value
           8A    D4    0331   783          CLRL    (R10)+                     ; mbz
                        0333   784 240$:
                        0333   785          ;
                        0333   786          ; TRM$_INISTRNG - initial data to pre-load
```

```
                     0333   787          ;
    52   30 AB   3C  0333   788          MOVZWL  CTP$W_SR_END_DATA(R11),R2 ; inistring + prompt
         52 56   C2  0337   789          SUBL    R6,R2                     ; minus prompt length
            21   13  033A   790          BEQL    250$                      ; Branch if no init string
         8A 52   B0  033C   791          MOVW    R2,(R10)+                 ; Length of init string
         8A 05   B0  033F   792          MOVW    #TRM$_INISTRNG,(R10)+     ; item code
         8A 51   D0  0342   793          MOVL    R1,(R10)+                 ; Address
            8A   D4  0345   794          CLRL    (R10)+                    ; mbz
         51 52   C0  0347   795          ADDL2   R2,R1                     ; Next string
                     034A   796          ;
                     034A   797          ; TRM$_INIOFFSET - where to echo inistring from
                     034A   798          ;
            8A   B4  034A   799          CLRW    (R10)+                    ; no value
         8A 08   B0  034C   800          MOVW    #TRM$_INIOFFSET,(R10)+    ; modifiers item code
            59   D5  034F   801          TSTL    R9                        ; zero?
            06   13  0351   802          BEQL    245$                      ; this parameter zero
    8A   59 56   C3  0353   803          SUBL3   R6,R9,(R10)+              ; store inioffset-prompt length
            02   11  0357   804          BRB     247$                      ; continue
            8A   D4  0359   805 245$:    CLRL    (R10)+                    ; zero parameter
            8A   D4  035B   806 247$:    CLRL    (R10)+                    ; mbz
                     035D   807 250$:
                     035D   808          ;
                     035D   809          ; TRM$_MODIFIERS
                     035D   810          ;
            8A   B4  035D   811          CLRW    (R10)+                    ; no value
         8A 00   B0  035F   812          MOVW    #TRM$_MODIFIERS,(R10)+    ; modifiers item code
    8A   58 3F   CB  0362   813          BICL3   #IO$M_FCODE,R8,(R10)+     ; CLEAR function code
    58 8031 8F   3C  0366   814          MOVZWL  #<IO$M_EXTEND!IO$_READVBLK>,R8 ; Set Item List QIO function
            8A   D4  036B   815          CLRL    (R10)+                    ; mbz
                     036D   816          ;
                     036D   817          ; TRM$_TERM - terminator mask
                     036D   818          ;
            54   D5  036D   819          TSTL    R4                        ; Use normal terminators if zero
            0C   13  036F   820          BEQL    260$                      ; Branch if so
       8A   20   B0  0371   821          MOVW    #32,(R10)+                ; length
       8A   03   B0  0374   822          MOVW    #TRM$_TERM,(R10)+         ; item code
    8A   08 A4   9E  0377   823          MOVAB   8(R4),(R10)+              ; address of terminator mask
            8A   D4  037B   824          CLRL    (R10)+                    ; mbz
                     037D   825 260$:
                     037D   826          ;
                     037D   827          ; Test for read verify
                     037D   828          ;
            11   91  037D   829          CMPB    #CTP$C_MT_VMS_READVFY,-
         2A 4B       037F   830                  CTP$B_MSGTYPE(R11)        ; see if read verify
            3D   12  0381   831          BNEQ    299$                      ; done with itemlist if not
```

```
                     0383   833          ;
                     0383   834          ; TRM$_EDITMODE - set read verify mode
                     0383   835          ;
          8A   B4    0383   836          CLRW    (R10)+                      ; no length
     8A   01   B0    0385   837          MOVW    #TRM$_EDITMODE,(R10)+       ; item code
     8A   01   9A    0388   838          MOVZBL  #TRM$K_EM_RDVERIFY,(R10)+ ; read verify
          8A   D4    038B   839          CLRL    (R10)+                      ; mbz
                     038D   840          ;
                     038D   841          ; TRM$_PICSTRNG - Picture string
                     038D   842          ;
52   3C AB   3C      038D   843          MOVZWL  CTP$W_SR2_PICSTRSIZE(R11),R2 ; length
          0E   13    0391   844          BEQL    270$                        ; Branch if zero
     8A   52   B0    0393   845          MOVW    R2,(R10)+                   ; length
     8A   06   B0    0396   846          MOVW    #TRM$_PICSTRNG,(R10)+       ; item code
     8A   51   D0    0399   847          MOVL    R1,(R10)+                   ; address
          8A   D4    039C   848          CLRL    (R10)+                      ; mbz
     51   52   C0    039E   849          ADDL    R2,R1                       ; Add length of picstring to address
                     03A1   850          ;
                     03A1   851          ; TRM$_ALTECHSTR - Alternate echo string
                     03A1   852          ;
                     03A1   853 270$:
52   3A AB   3C      03A1   854          MOVZWL  CTP$W_SR2_ALTECHSIZE(R11),R2 ; length
          0B   13    03A5   855          BEQL    280$                        ; Branch if zero
     8A   52   B0    03A7   856          MOVW    R2,(R10)+                   ; length
     8A   09   B0    03AA   857          MOVW    #TRM$_ALTECHSTR,(R10)+      ; item code
     8A   51   D0    03AD   858          MOVL    R1,(R10)+                   ; address
          8A   D4    03B0   859          CLRL    (R10)+                      ; mbz
                     03B2   860 280$:
                     03B2   861          ;
                     03B2   862          ; TRM$_FILLCHR - Fill characters
                     03B2   863          ;
52   40 AB   3C      03B2   864          MOVZWL  CTP$W_SR2_FILLCHAR(R11),R2 ; fetch fill characters
                     03B6   865 ;***      BEQL    285$                        ; branch if none
          8A   B4    03B6   866          CLRW    (R10)+                      ; zero length
     8A   07   B0    03B8   867          MOVW    #TRM$_FILLCHR ,(R10)+       ; item code
     8A   52   D0    03BB   868          MOVL    R2,(R10)+                   ; value
          8A   D4    03BE   869          CLRL    (R10)+                      ; mbz
                     03C0   870          ;
                     03C0   871          ; Done with read verify items
                     03C0   872          ;
                     03C0   873 285$:
```

```
                                03C0    875 299$:   SUBL3   R5,R10,R6               ; Length of item list
                             53 7C      03C4    876         CLRQ    R3                      ; no P3,P4
                                        03C6    877 300$:
                                        03C6    878         ;
                                        03C6    879         ; Allocate read buffer
                                        03C6    880         ;
                          52   2E AB    B0      03C6    881         MOVW    CTP$W_SR_MAX_LEN(R11),R2   ; Length of read
                                50    DD      03CA    882         PUSHL   R0                      ; Save main buffer
                              FC31'   30      03CC    883         BSBW    GETBUF                  ; Get input buffer
                             51   50    D0      03CF    884         MOVL    R0,R1                   ; Set address
                                50 8ED0      03D2    885         POPL    R0                      ; Restore main buffer
                          12 A0 51    D0      03D5    886         MOVL    R1,AST$L_ODATA(R0)      ; save input buffer address
                          51   32 A1 9E      03D9    887         MOVAB   CTP$T_RD_DATA(R1),R1    ; Set address for returned data
                                        03DD    888
                          00000000'EF   95      03DD    889         TSTB    INDFLAG                 ; Indirect file processing?
                                03    12      03E3    890         BNEQ    320$                    ; Branch if yes
                                        03E5    891 310$:
                                027D    31      03E5    892         BRW     CTERM_PROCMSG_QIO          ; Do QIO and Exit
                                        03E8    893
                                        03E8    894 320$:
                                53    DD      03E8    895         PUSHL   R3                      ; Save
                             53 52    D0      03EA    896         MOVL    R2,R3                   ; Set length
                              FC10'   30      03ED    897         BSBW    VMS_INDREAD             ; read from file
                                53 8ED0      03F0    898         POPL    R3                      ; If code path returns here,
                                F0    11      03F3    899         BRB     310$                    ;  routine got EOF, continue with QIO
                                        03F5    900
```

```
                     03F5   902 UNREAD_MSG:                                ; message type = 5
                     03F5   903
           16   10   03F5   904          BSBB    CANCEL_READ               ; cancel read
        02E5    31   03F7   905          BRW     CTERM_PROCMSG_EXIT        ; Exit back to common code
                     03FA   906
                     03FA   907 C_R_INPUT_MSG:                            ; message type = 6
                     03FA   908 ;
                     03FA   909 ; *** note that the following code path will never be taken
                     03FA   910 ; *** when talking VAX to VAX.
                     03FA   911 ;
           11   10   03FA   912          BSBB    CANCEL_READ               ; cancel read
   58   08B1 8F    3C   03FC   913          MOVZWL  #IO$_READVBLK!IO$M_PURGE!IO$M_TIMED,R8 ; Purge typeahead
     51    2C AB    9E   0401   914          MOVAB   CTP$B_CI_FLAGS+1(RT1),R1 ; Buffer addr
   52   0100 8F    3C   0405   915          MOVZWL  #256,R2                   ; a random positive number ...
        0258    31   040A   916          BRW     CTERM_PROCMSG_QIO         ; Exit back to common code
                     040D   917 ;
                     040D   918 ; Local routine to cancel active read
                     040D   919 ;
                     040D   920 CANCEL_READ:
           50   DD   040D   921          PUSHL   R0                        ; save
                     040F   922          $CANCEL_S CHAN = READCHAN          ; Cancel the read
                     041D   923          ONERROR QUIT                      ; quit on error
           50 8ED0   0445   924          POPL    R0                        ; restore
                05   0448   925          RSB                               ; return
```

```
                                0449    927 WRITE_MSG:                                  ; message type = 7
                                0449    928
                00000000'EF  D6 0449    929         INCL    WRITEQIO                    ; Set "write active"
                                044F    930
            57  00000000'EF  B0 044F    931         MOVW    WRITECHAN,R7                ; Set write channel
                    58   30  9A 0456    932         MOVZBL  #IO$_WRITEVBLK,R8           ; Set function code
                                0459    933         ;
                                0459    934         ; Must use flags from message with BEGIN set.
                                0459    935         ;
                59  017B'CF   9E 0459    936         MOVAB   W^SAVE_WR_FLAGS,R9          ; set address
                  5A    2B AB 3C 045E    937         MOVZWL  CTP$W_WR_FLAGS(R11),R10     ; Flags
                  03 5A   04  E1 0462    938         BBC     #CTP$V_WR_BEGIN,R10,5$      ; Branch if not BEGIN
                     69   5A  B0 0466    939         MOVW    R10,(R9)                    ; Save
                                0469    940 5$:
                  03 5A   05  E1 0469    941         BBC     #CTP$V_WR_END,R10,7$        ; Branch if not END
                     69   20  A8 046D    942         BISW    #CTP$M_WR_END,(R9)          ; Set end in original flags
                                0470    943 7$:
                     5A   69  B0 0470    944         MOVW    (R9),R10                    ; Pick up flags for this write
                     69   10  AA 0473    945         BICW    #CTP$M_WR_BEGIN,(R9)        ; Clear BEGIN in original flags
                                0476    946
                  51    2F AB 9E 0476    947         MOVAB   CTP$T_WR_DATA(R11),R1       ; Set address of write data
                              05 A3 047A    948         SUBW3   #<CTP$T_WR_DATA-CTP$B_MSGTYPE>,-; Compute length of write
                  52    28 AB    047C    949                 CTP$W_MSGSIZE(R11),R2       ;  as length of message minus overhe
                  08 5A   02  E1 047F    950         BBC     #CTP$V_WR_NEWLINE,R10,10$   ; newline is VMS specific
                                0483    951
                58  0400 8F   A8 0483    952         BISW    #IO$M_NEWLINE,R8            ; set newline
                       01DA   31 0488    953         BRW     CTERM_PROCMSG_QIO           ; done... (no logging)
                                048B    954 ;
                                048B    955 ; Compute carriage control and refresh
                                048B    956 ;
                                048B    957 10$:
                          00    EF 048B    958         EXTZV   #CTP$V_WR_LOCK,-
                59  5A    02       048D    959                 #CTP$S_WR_LOCK,R10,R9       ; Extract locking field
                                0490    960         ;
                                0490    961         ; it is assumed here that all writes happen with locking
                                0490    962         ;
                     59   03  91 0490    963         CMPB    #CTP$M_WR_BEFAFTRE,R9       ; redisplay set?
                          05  12 0493    964         BNEQ    15$                         ; branch if not
                58   2000 8F   A8 0495    965         BISW    #IO$M_REFRESH,R8            ; Set refresh
                                049A    966 15$:
                  17 5A   04  E1 049A    967         BBC     #CTP$V_WR_BEGIN,R10,20$     ; Not beginning of message, skip flags
                       2D AB  F0 049E    968         INSV    CTP$B_WR_PREFIX(R11),-
                54   08   10       04A1    969                 #16,#8,R4                   ; Insert prefix into high word
                       2E AB  90 04A4    970         MOVB    CTP$B_WR_POSTFIX(R11),-
                     017A'CF       04A7    971                 W^SAVE_WR_POSTFIX           ; Save postfix for later
                     5A   07  E1 04AA    972         BBC     #CTP$V_WR_PREFIX+1, -
                          07       04AD    973                 R10,20$                     ; *** hack. branch if not char
            54  00800000 8F   C8 04AE    974         BISL    #^X80@16,R4                 ; Set character bit
                                04B5    975
                  12 5A   05  E1 04B5    976 20$:    BBC     #CTP$V_WR_END,R10,30$       ; Skip if not end
                     017A'CF   F0 04B9    977         INSV    W^SAVE_WR_POSTFIX,-
                54   08   18       04BD    978                 #24,#8,R4                   ; Into high byte
                     5A   09  E1 04C0    979         BBC     #CTP$V_WR_POSTFIX+1, -
                          07       04C3    980                 R10,30$                     ; *** hack. branch if not char
            54  80000000 8F   C8 04C4    981         BISL    #^X80@<16+8>,R4             ; Set character bit
                                04CB    982 30$:
                          03  E1 04CB    983         BBC     #CTP$V_WR_DISCARD,-
```

```
                 0C 5A        04CD   984                     R10,40$                 ; Cancel Discard state?
          58     0040 8F  AB  04CF   985             BISW    #IO$M_CANCTRLO,R8       ; Set cancel control O
                          AA  04D4   986             BICW    #<FLG$M_CTRL_O!-
                              04D5   987                     FLG$M_CTRL_C!>,-
00000000'EF       06        04D5   988                     CTERM_FLAG              ; Clear ^O and flushing
                            04DB   989 40$:
                 0B   E1     04DB   990             BBC     #CTP$V_WR_TRANSPARENT,-
                 05 5A       04DD   991                     R10,50$                 ; Write transparent?
          58     0100 8F  A8  04DF   992             BISW    #IO$M_NOFORMAT,R8       ; Set write no format
                            04E4   993 50$:
 00000000'EF       D5        04E4   994             TSTL    READQIO                 ; read active?
                   05   13   04EA   995             BEQL    55$                     ; skip if not
          58     0200 8F  A8  04EC   996             BISW    #IO$M_BREAKTHRU,R8      ; set breakthru (fake full duplex)
                            04F1   997 55$:
                   03   E0   04F1   998             BBS     #FLG$V_LOGGING,-
            03 0000'CF        04F3   999                     W^CTERM_FLAG,60$        ; Branch if logging
                 016B   31   04F7  1000             BRW     CTERM_PROCMSG_QIO       ; Exit back to common code
                            04FA  1001 60$:
                FB03'   30   04FA  1002             BSBW    CTERM$LOG_IO            ; ** LOG WRITE
                 0165   31   04FD  1003             BRW     CTERM_PROCMSG_QIO       ; Exit back to common code
                            0500  1004
```

```
                          0500  1006 READ_CHAR_MSG:                          ; message type = 10
                          0500  1007
          00000000'EF  D6 0500  1008          INCL    READQIO                 ; Set "read active"
                          0506  1009
              58   27  90 0506  1010          MOVW    #IO$_SENSEMODE,R8       ; Set qio function
          51  00000000'EF 9E 0509  1011          MOVAB   CHAR_BLOCK,R1           ; Characteristics block
              52   0C  90 0510  1012          MOVB    #12,R2                  ; Length
                          0513  1013
                  014F  31 0513  1014          BRW     CTERM_PROCMSG_QIO       ; Exit back to common code
                          0516  1015
```

```
                        0516  1017 CHAR_MSG:                                ; message type = 11
                        0516  1018
    00000000'EF   D6    0516  1019          INCL    READQIO                 ; Set "read active"
                        051C  1020
        5E   08   C2    051C  1021          SUBL2   #8,SP                   ; allocate local area
        5A   5E   D0    051F  1022          MOVL    SP,R10                  ; Set address of 8 byte area
             6A   7C    0522  1023          CLRQ    (R10)                   ; Clear
59  00000000'EF   9E    0524  1024          MOVAB   CHAR_BLOCK,R9           ; Address of characteristics block
                        052B  1025
        0FFF 8F   BB    052B  1026          PUSHR   #^M<R0,R1,R2,R3,R4,R5, -
                        052F  1027                  R6,R7,R8,R9,R10,R11>
        52   5B   D0    052F  1028          MOVL    R11,R2                  ; Set address of CTP
                FACB'   30    0532  1029          BSBW    CT_CHAR_MSG             ; Map cterm list into VMS
                        0535  1030 ;
                        0535  1031 ; Check for out of band
                        0535  1032 ;
56  00000107'EF   9E    0535  1033          MOVAB   OUTBAND_SET,R6          ; Currently enabled
57  00000127'EF   9E    053C  1034          MOVAB   OUTBAND_NEW,R7          ; As returned by routine above
                        0543  1035
    67   66   20   29    0543  1036          CMPC3   #OOB_LEN,(R6),(R7)      ; See if it changed
              03   12    0547  1037          BNEQ    10$                     ; Branch if equal
            00F1   31    0549  1038          BRW     100$                    ; Branch if not
                        054C  1039 10$:
          1C A7   CA    054C  1040          BICL    OOB_ECHO(R7),-
          14 A7         054F  1041                  OOB_ABORT(R7)           ; Clear normal ^C, ^Y AST in OOB abort
                        0551  1042 ;***     BICL    #<1@TTY$C_CTRLY>,-
                        0551  1043 ;***             OOB_ABORT(R7)           ; Clear ^Y abort, always enabled
                        0551  1044 ;***     BBC     #TTY$C_CTRLC,-
                        0551  1045 ;***             OOB_ABORT(R7),15$       ; Branch if no ctrl/c
                        0551  1046 ;***     BBC     #TTY$C_CTRLC,-
                        0551  1047 ;***             OOB_ECHO(R7),15$        ; Branch if no echo on ctrl/c
                        0551  1048 ;***     BICL    #<1@TTY$C_CTRLC>,-
                        0551  1049 ;***             OOB_ABORT(R7)           ; Clear ^C in abort
                        0551  1050 ;***     BISW    #FLG$M_CTRLC,W^CTERM_FLAG ; set: enable standard ^C
                        0551  1051
                        0551  1052 15$:
                        0551  1053 ;
                        0551  1054 ; Enable out-of-band includes
                        0551  1055 ;
        0C A6   D1      0551  1056          CMPL    OOB_INCLUDE(R6),-
        0C A7           0554  1057                  OOB_INCLUDE(R7)         ; Compare include sets
             29   13    0556  1058          BEQL    20$                     ; branch if no change
50      08 A7   9E      0558  1059          MOVAB   OOB_INCLUDE-4(R7),R0    ; Set address of mask descriptor
                        055C  1060          $QIO_S  CHAN = OUTBANDINC,-
                        055C  1061                  FUNC = #IO$_SETMODE!IO$M_OUTBAND!IO$M_INCLUDE,-
                        055C  1062                  P1   = CTERM_OUTBANDAST,-
                        055C  1063                  P2   = R0
                        0581  1064 ;
                        0581  1065 ; Enable out-of-band excludes
                        0581  1066 ;
                        0581  1067 20$:
        04 A6   D1      0581  1068          CMPL    OOB_EXCLUDE(R6),-
        04 A7           0584  1069                  OOB_EXCLUDE(R7)         ; Compare exclude sets
             28   13    0586  1070          BEQL    30$                     ; branch if no change
   50    67   9E      0588  1071          MOVAB   OOB_EXCLUDE-4(R7),R0    ; Set address of mask descriptor
                        058B  1072
                        058B  1073          $QIO_S  CHAN = OUTBANDEXC,-
```

```
                     058B  1074                    FUNC = #IO$_SETMODE!IO$M_OUTBAND,-
                     058B  1075                    P1   = CTERM_OUTBANDAST,-
                     058B  1076                    P2   = R0
                     05B0  1077 ;
                     05B0  1078 ; Enable out of band aborts
                     05B0  1079 ;
                     05B0  1080 30$:
          14 A6   D1 05B0  1081            CMPL    OOB_ABORT(R6),-
          14 A7       05B3  1082                    OOB_ABORT(R7)           ; Compare ABORT sets
             29   13 05B5  1083            BEQL    40$                     ; branch if no change
       50 10 A7   9E 05B7  1084            MOVAB   OOB_ABORT-4(R7),R0      ; Set address of mask descriptor
                     05BB  1085            $QIO_S  CHAN = OUTBANDABO,-
                     05BB  1086                    FUNC = #IO$_SETMODE!IO$M_OUTBAND!IO$M_TT_ABORT,-
                     05BB  1087                    P1   = CTERM_OUTBANDAST,-
                     05BB  1088                    P2   = R0
                     05E0  1089 40$:
    66   67   20   28 05E0  1090            MOVC3   #OOB_LEN,(R7),(R6)      ; Copy new set
                     05E4  1091
             03   E1 05E4  1092            BBC     #TTY$C_CTRLC,-
          1C 1C A7       05E6  1093                    OOB_ECHO(R7),80$        ; Branch if ^C is not to be enabled
    00000000'EF   95 05E9  1094            TSTB    CNTRCFLAG               ; See if enabled
             45   12 05EF  1095            BNEQ    95$                     ; yes, exit
                     05F1  1096 ;
                     05F1  1097 ; Enable ^C
                     05F1  1098 ;
    00000000'EF   96 05F1  1099            INCB    CNTRCFLAG               ; set enabled
50  00000000'EF   9E 05F7  1100            MOVAB   CNTRLC_AST,R0           ; Address of AST routine
51    0100 8F   3C 05FE  1101            MOVZWL  #IO$M_CTRLCAST,R1       ; Parameter
             10   11 0603  1102            BRB     90$
                     0605  1103 ;
                     0605  1104 ; Disable ^C
                     0605  1105 ;
                     0605  1106 80$:
00000000'EF   95 0605  1107            TSTB    CNTRCFLAG               ; See if disabled
          29   13 060B  1108            BEQL    95$                     ; yes, exit
00000000'EF   94 060D  1109            CLRB    CNTRCFLAG               ; set disabled
          50   7C 0613  1110            CLRQ    R0                      ; Set inputs for qio
                     0615  1111 90$:
                     0615  1112            $QIO_S  CHAN = CNTRLCHAN,-       ; Control C channel
                     0615  1113                    FUNC = #<IO$_SETMODE!IO$M_CTRLCAST>,-
                     0615  1114                    P1   = (R0),-            ; Ast routine
                     0615  1115                    P2   = R1                ; ast parameter
                     0636  1116 95$:
    0FFF 8F   BA 0636  1117            POPR    #^M<R0,R1,R2,R3,R4,R5, -
                     063A  1118                    R6,R7,R8,R9,R10,R11>
       00A2   31 063A  1119            BRW     CTERM_PROCMSG_EXIT      ; Exit back to common code
                     063D  1120
                     063D  1121 100$:
    0FFF 8F   BA 063D  1122            POPR    #^M<R0,R1,R2,R3,R4,R5, -
                     0641  1123                    R6,R7,R8,R9,R10,R11>
                     0641  1124 ;
                     0641  1125 ; NOTE***: note that there is a bug doing a set host/cterm
                     0641  1126 ; followed by a  set host/old. in logging off, an enable ^T gets here
                     0641  1127 ; in the code...???
                     0641  1128 ;
    51   59   D0 0641  1129            MOVL    R9,R1                   ; Set address of buffer
    52   0C   90 0644  1130            MOVB    #12,R2                  ; Set length
```

```
                53   02 AA    B0  0647  1131          MOVW    2(R10),R3               ; Set speed
                54   04 AA    B0  064B  1132          MOVW    4(R10),R4               ; Set fill
                55   06 AA    90  064F  1133          MOVB    6(R10),R5               ; Set parity
                   58   23    B0  0653  1134          MOVW    #IO$_SETMODE,R8         ; Set function code
                                0656  1135
                        000C    31  0656  1136          BRW     CTERM_PROCMSG_QIO       ; Do Qio and exit
                                0659  1137
                                0659  1138 CHECK_INP_MSG:                            ; message type = 12
                                0659  1139
             58    0067 8F    3C  0659  1140          MOVZWL  #IO$_SENSEMODE!IO$M_TYPEAHDCNT,R8       ; function code
               51    2C AB    9E  065E  1141          MOVAB   CTP$Q_IC_COUNT(R11),R1  ; Address
                                0662  1142
                        0000    31  0662  1143          BRW     CTERM_PROCMSG_QIO       ; go to common code
                                0665  1144
                                0665  1145 CTERM_PROCMSG_QIO:                        ; Inputs are now ready to do QIO (R0-R8)
                                0665  1146
                           01   E0  0665  1147          BBS     #FLG$V_CTRL_CY,-
                 6B 0000'CF         0667  1148                  W^CTERM_FLAG,10$         ; Branch if flushing due to ^C or ^Y
                           50   DD  066B  1149          PUSHL   R0                      ; Save in case of QIO error
                                066D  1150
                      7E   55   7D  066D  1151          MOVQ    R5,-(SP)                ; P5,P6
                      7E   53   7D  0670  1152          MOVQ    R3,-(SP)                ; P3,P4
                      7E   51   7D  0673  1153          MOVQ    R1,-(SP)                ; P1,P2
                           50   DD  0676  1154          PUSHL   R0                      ; ASTPRM
                    0000'CF   9F  0678  1155          PUSHAB  W^ASTHANDLER            ; ASTADR
                      04 A0   7F  067C  1156          PUSHAQ  AST$Q_IOSB(R0)          ; IOSB
                      7E   57   7D  067F  1157          MOVQ    R7,-(SP)                ; CHAN,FUNC
                           7E   D4  0682  1158          CLRL    -(SP)                   ; EFN
          00000000'GF      0C   FB  0684  1159          CALLS   #12,G^SYS$QIO           ; call qio
                                068B  1160          IF_NO_QUOTA_QUIT                ; In quota error, quit
                        01 50   E9  0685  1161          BLBC    R0,5$                   ; If error, handle ast manually
                                04  0688  1162          RET
                                0689  1163 5$:
                      52   50   D0  0689  1164          MOVL    R0,R2                   ; save status
                           50   DD  068C  1165          PUSHL   R0                      ; push status from qio
                           7E   D4  068E  1166          CLRL    -(SP)                   ; no fao
                00000000'8F     DD  06C0  1167          PUSHL   #REM$_QIOERR            ; qio error
          00000000'GF      03   FB  06C6  1168          CALLS   #3,G^LIB$SIGNAL         ; signal
                           50 8ED0  06CD  1169          POPL    R0                      ; Restore buffer address
                 04 A0     52   D0  06D0  1170          MOVL    R2,AST$Q_IOSB(R0)       ; store in IOSB
                           06   11  06D4  1171          BRB     20$                     ; Go to declare ast
                                06D6  1172
                    0176'CF     D0  06D6  1173 10$:     MOVL    W^FLUSH_STATUS,-
                      04 A0             06DA  1174                  AST$Q_IOSB(R0)          ; Set status
                      08 A0     D4  06DC  1175 20$:     CLRL    AST$Q_IOSB+4(R0)        ; Set status
                                06DF  1176                                              ; fall through to DCLAST
                                06DF  1177
                                06DF  1178 CTERM_PROCMSG_EXIT:
                                06DF  1179
                                06DF  1180          $DCLAST_S ASTADR = ASTHANDLER,-
                                06DF  1181                    ASTPRM = R0           ; CALL AST
                                06F0  1182
                                04  06F0  1183          RET
                                06F1  1184
```

```
                     06F1  1186          .SBTTL  CTERM_QIODONE - Process a completed terminal QIO
                     06F1  1187
                     06F1  1188 ;++
                     06F1  1189 ;
                     06F1  1190 ; FUNCTIONAL DESCRIPTION:
                     06F1  1191 ;
                     06F1  1192 ;
                     06F1  1193 ; CALLING SEQUENCE:
                     06F1  1194 ;
                     06F1  1195 ;       BSBW    CTERM_QIODONE
                     06F1  1196 ;
                     06F1  1197 ; INPUT PARAMETERS:
                     06F1  1198 ;
                     06F1  1199 ;       R0 - points to AST control block
                     06F1  1200 ;
                     06F1  1201 ; IMPLICIT INPUTS:
                     06F1  1202 ;       NONE
                     06F1  1203 ;
                     06F1  1204 ; OUTPUT PARAMETERS:
                     06F1  1205 ;
                     06F1  1206 ;       R0 = 0 means no message to send to link.
                     06F1  1207 ;       R0 > 0 means send completion message to link.
                     06F1  1208 ;       If R0 > 0 Then
                     06F1  1209 ;       R1 - Length of message to write to net
                     06F1  1210 ;       R2 - Address of message to write to net
                     06F1  1211 ;
                     06F1  1212 ; IMPLICIT OUTPUTS:
                     06F1  1213 ;
                     06F1  1214 ;       RETSTATUS
                     06F1  1215 ;
                     06F1  1216 ; COMPLETION CODES:
                     06F1  1217 ;       NONE
                     06F1  1218 ;
                     06F1  1219 ; SIDE EFFECTS:
                     06F1  1220 ;
                     06F1  1221 ;       AST control block may be returned to free list.
                     06F1  1222 ;       R3 destroyed.
                     06F1  1223 ;
                     06F1  1224 ;--
                     06F1  1225
                     06F1  1226 ;+
                     06F1  1227 ;
                     06F1  1228 ; Valid message types here are:
                     06F1  1229 ;
                     06F1  1230 ; ctp$c_mt_start_rd     (2)     Start Read              (H  ---> S)
                     06F1  1231 ; ctp$c_mt_unread       (5)     Unread                  (H  ---> S)
                     06F1  1232 ; ctp$c_mt_clr_input    (6)     Clear Input             (H  ---> S)
                     06F1  1233 ; ctp$c_mt_write        (7)     Write                   (H  ---> S)
                     06F1  1234 ; ctp$c_mt_read_char    (10)    Read Characteristics    (H  ---> S)
                     06F1  1235 ; ctp$c_mt_char         (11)    Characteristics         (H <---> S)
                     06F1  1236 ; ctp$c_mt_check_inp    (12)    Check Input             (H  ---> S)
                     06F1  1237 ;
                     06F1  1238 ;-
                     06F1  1239
                     06F1  1240 CTERM_QIODONE::
                     06F1  1241
5B   0E A0   3C      06F1  1242          MOVZWL  AST$W_OFFSET(R0),R11    ; Offset to current message
```

```
                                                     H 9
                5B    50    C0  06F5  1243          ADDL2   R0,R11                  ; Add in base
                                06F8  1244
      60  00000000'EF    9E  06F8  1245          MOVAB   LNKWRTDONE,AST$L_STATE(R0)      ; Set next state
                      53    D4  06FF  1246          CLRL    R3                      ; Assume nothing to write to net
                                0701  1247
                                0701  1248          CASE    CTP$B_MSGTYPE(R11),-    ; Case on message type
                                0701  1249                  <10$,-                  ; 0
                                0701  1250                  INIT_DONE,-             ; 1
                                0701  1251                  START_READ_DONE,-       ; 2
                                0701  1252                  10$,10$,-               ; 3,4
                                0701  1253                  UNREAD_DONE,-           ; 5
                                0701  1254                  CLR_INPUT_DONE,-        ; 6
                                0701  1255                  WRITE_DONE,-            ; 7
                                0701  1256                  10$,10$,-               ; 8,9
                                0701  1257                  READ_CHAR_DONE,-        ; 10
                                0701  1258                  CHAR_DONE,-             ; 11
                                0701  1259                  CHECK_INP_DONE,-        ; 12
                                0701  1260                  10$,10$,-               ; 13,14
                                0701  1261                  VMSQIO_DONE,-           ; 15 is VMS QIO
                                0701  1262                  10$,START_READ_DONE-    ; 16,17 (read verify)
                                0701  1263                  >,-
                                0701  1264                  TYPE = B                ; Byte field
                                072A  1265 ;
                                072A  1266 ; Shouldn't get to here
                                072A  1267 ;
                                072A  1268 10$:
                                072A  1269          QUIT    #REM$_INTERR            ; internal error
                          05    0753  1270          RSB
                                0754  1271
                                0754  1272 ;
                                0754  1273 ; message completion
                                0754  1274 ;
                                0754  1275
                                0754  1276 VMSQIO_DONE:
                F8A9'     30    0754  1277          BSBW    CT_VMSQIO_DONE          ; set up return data if necessary
                0179      31    0757  1278          BRW     CTERM_QIODONE_EXIT      ; Exit back to common code
                                075A  1279
                                075A  1280 INIT_DONE:
                0176      31    075A  1281          BRW     CTERM_QIODONE_EXIT      ; Exit back to common code
                                075D  1282
                                075D  1283 START_READ_DONE:                       ; message type = 2
                                075D  1284
00000000'EF       D7    075D  1285          DECL    READQIO                 ; Clear read active
                                0763  1286
                   50   DD    0763  1287          PUSHL   R0                      ; Save main buffer
          50    16 A0   D0    0765  1288          MOVL    AST$L_ITMLST(R0),R0     ; See if there was an itemlist
                   03   13    0769  1289          BEQL    5$                      ; Branch if not
                F892'   30    076B  1290          BSBW    BUFFREE                 ; Free buffer
                   50 8ED0    076E  1291 5$:      POPL    R0                      ; Restore buffer
          53    12 A0   D0    0771  1292          MOVL    AST$L_ODATA(R0),R3      ; Get return buffer address
                                0775  1293          ;
                                0775  1294          ; Fill in return READ_DATA buffer
                                0775  1295          ;
          26 A3   09    9B    0775  1296          MOVZBW  #PRO$C_DATA, -
                                0779  1297                  CTP$B_PRO_MSGTYPE(R3)   ; Set foundation data message
               2A A3    7C    0779  1298          CLRQ    CTP$B_MSGTYPE(R3)       ; Clear header up to data
                   03   90    077C  1299          MOVB    #CTP$C_MT_READ_DATA,-
```

```
                2A A3          077E  1300                    CTP$B_MSGTYPE(R3)       ; Set message type
          51    06 A0    3C    0780  1301            MOVZWL  AST$Q_IOSB+2(R0),R1     ; Get length of read up to terminator
          30 A3    51    B0    0784  1302            MOVW    R1,CTP$W_RD_TERM_POS(R3); Set terminator position
                0B A0    90    0788  1303            MOVB    AST$Q_IOSB+7(R0),-
                2F A3          078B  1304                    CTP$B_RD_CURS_POS(R3)   ; Fetch cursor offset from EOL
                               078D  1305
          52    0A A0    9A    078D  1306            MOVZBL  AST$Q_IOSB+6(R0),R2     ; Get terminator length
             51    52    A0    0791  1307            ADDW    R2,R1                   ; Get total length of read data
                               0794  1308            ;
                               0794  1309            ; Write logging file
                               0794  1310            ;
                      03       E1    0794  1311            BBC     #FLG$V_LOGGING,-
          22 0000'CF                 0796  1312                    W^CTERM_FLAG,20$        ; Branch if not logging
                      3F       BB    079A  1313            PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save
             50    32 A3       9E    079C  1314            MOVAB   CTP$T_RD_DATA(R3),R0    ; address of returned data
          53    05C1'CF       D0    07A0  1315            MOVL    W^LOG_ADDR,R3           ; set up address
                      0B       E0    07A5  1316            BBS     #CTP$V_SR_NOECHO,-
             04 2B AB                07A7  1317                    CTP$L_SR_FLAGS(R11),10$ ; do not log input in NOECHO
          63    60    51    28    07AA  1318            MOVC3   R1,(R0),(R3)            ; copy data in after prompt
          51    01A7'CF       9E    07AE  1319  10$:      MOVAB   W^LOG_BUF,R1            ; address
          52    53    51    C3    07B3  1320            SUBL3   R1,R3,R2                ; length
                   F846'      30    07B7  1321            BSBW    CTERM$LOG_IO            ; log data
                      3F       BA    07BA  1322            POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore
                               07BC  1323  20$:
                51    08       C0    07BC  1324            ADDL    #CTP$C_RD_PROLEN,R1     ; Add in buffer header length
                               07BF  1325  ;
                               07BF  1326  ; Map status
                               07BF  1327  ;
       55    00000147'EF       9E    07BF  1328            MOVAB   RD_STAT_TBL,R5          ; Get read status table
       54    0000016F'EF       9E    07C6  1329            MOVAB   RD_STAT_END,R4          ; Get end of table
                               07CD  1330  50$:
             04 A0    85       B1    07CD  1331            CMPW    (R5)+,AST$Q_IOSB(R0)    ; Compare
                      07       13    07D1  1332            BEQL    60$                     ; Branch on match
                      85       B5    07D3  1333            TSTW    (R5)+                   ; bump pointer
                55    54       D1    07D5  1334            CMPL    R4,R5                   ; End of table?
                      F3       14    07D8  1335            BGTR    50$                     ; yes, status not in table
                               07DA  1336  60$:
          2B A3       65       90    07DA  1337            MOVB    (R5),CTP$B_RD_FLAGS(R3) ; Set status
            25 04 A0    E9       07DE  1338            BLBC    AST$Q_IOSB(R0),70$      ; Continue if error
                               07E2  1339            ;
                               07E2  1340            ; Check for "valid escape" or "buffer overflow",
                               07E2  1341            ;  both returned as SS$_NORMAL
                               07E2  1342            ;
                      1B       91    07E2  1343            CMPB    #TTY$C_ESCAPE,-
                08 A0                  07E4  1344                    AST$Q_IOSB+4(R0)        ; Escape?
                      10       12    07E6  1345            BNEQ    65$                     ; Branch if not
             1B 04 A0    E9       07E8  1346            BLBC    AST$Q_IOSB(R0),70$      ; other error
          0A A0    01       91    07EC  1347            CMPB    #1,AST$Q_IOSB+6(R0)     ; escape sequence?
                      15       13    07F0  1348            BEQL    70$                     ; Branch if not
                      01       90    07F2  1349            MOVB    #CTP$M_RD_VALESC,-
                2B A3                  07F4  1350                    CTP$B_RD_FLAGS(R3)      ; Set "valid escape"
                      0F       11    07F6  1351            BRB     70$                     ; Continue
                               07F8  1352  65$:
          04 A0    01       B1    07F8  1353            CMPW    #SS$_NORMAL,AST$Q_IOSB(R0) ; make sure wasn't ^C/^Y
                      09       12    07FC  1354            BNEQ    70$                     ; branch if it was
                08 A0    95       07FE  1355            TSTB    AST$Q_IOSB+4(R0)        ; Was there a terminator at all?
                      04       12    0801  1356            BNEQ    70$                     ; branch if yes
```

```
                04   90  0803  1357          MOVB    #CTP$M_RD_INPFULL,-
            2B A3        0805  1358                  CTP$B_RD_FLAGS(R3)      ; Set "input full"
                         0807  1359 ;
                         0807  1360 ; Set up for write to net
                         0807  1361 ;
                         0807  1362 70$:
                04   E1  0807  1363          BBC     #FLG$V_VAXHOST,-
      07 0000'CF         0809  1364                  W^CTERM_FLAG,80$         ; Branch if not VAX to VAX
      52    26 A3  9E    080D  1365 75$:     MOVAB   CTP$B_PRO_MSGTYPE(R3),R2 ; Address of message
            00BF   31    0811  1366          BRW     CTERM_QIODONE_EXIT      ; Exit back to common code
                         0814  1367 ;
                         0814  1368 ; Check for more data in typeahead.
                         0814  1369 ;
                         0814  1370 80$:
          7E    50   7D  0814  1371          MOVQ    R0,-(SP)                ; save
    0B6D'CF     00   FB  0817  1372          CALLS   #0,W^CHECK_MOREDATA     ; get status
            04   50   F0  081C  1373          INSV    R0,#CTP$V_RD_MOR_DATA,-
        2B A3   01       081F  1374                  #1,CTP$B_RD_FLAGS(R3)   ; insert bit
            50   8E   7D  0822  1375          MOVQ    (SP)+,R0                ; restore
                 E6   11  0825  1376          BRB     75$                     ; and go back to normal path
                         0827  1377
                         0827  1378 UNREAD_DONE:                             ; message type = 5
                         0827  1379
                 00A9    31  0827  1380          BRW     CTERM_QIODONE_EXIT      ; Exit back to common code
                         082A  1381
                         082A  1382 CLR_INPUT_DONE:                          ; message type = 6
                         082A  1383
                 00A6    31  082A  1384          BRW     CTERM_QIODONE_EXIT      ; Exit back to common code
                         082D  1385
                         082D  1386 WRITE_DONE:                              ; message type = 7
                         082D  1387
00000000'EF         D7   082D  1388          DECL    WRITEQIO                ; Clear write active
                         0833  1389
                    0A   E0  0833  1390          BBS     #CTP$V_WR_STATUS,-
          03 2B AB           0835  1391                  CTP$W_WR_FLAGS(R11),10$ ; Continue if status requested.
                 0098    31  0838  1392          BRW     CTERM_QIODONE_EXIT      ; Exit back to common code
                         083B  1393 ;
                         083B  1394 ; Fetch a WRITE COMPLETE packet to write back to net with
                         083B  1395 ;
                         083B  1396 10$:
                    50   DD  083B  1397          PUSHL   R0                      ; Save current work buffer
                  F7C0'  30  083D  1398          BSBW    GETBUF                  ; Get a buffer
              53    50   D0  0840  1399          MOVL    R0,R3                   ; Save new buffer address
                    50 8ED0  0843  1400          POPL    R0                      ; Restore current work buffer
                         0846  1401
          52    26 A3  9E    0846  1402          MOVAB   CTP$B_PRO_MSGTYPE(R3),R2 ; Address of message
              51    06   3C  084A  1403          MOVZWL  #CTP$C_WC_PROLEN,R1     ; Length of message
                         084D  1404
          26 A3   09   9B  084D  1405          MOVZBW  #PRO$C_DATA, -
                         0851  1406                  CTP$B_PRO_MSGTYPE(R3)   ; Set foundation data message
                    C8   9B  0851  1407          MOVZBW  #CTP$C_MT_WRITE_COM,-
                2A A3        0853  1408                  CTP$B_MSGTYPE(R3)       ; set message type, zero flags
                08 A0   D0   0855  1409          MOVL    AST$Q_IOSB+4(R0),-
                2C A3        0858  1410                  CTP$W_WC_HORPOS(R3)     ; Set horizontal and vertical position
                         085A  1411 ;
                         085A  1412 ; **** STATUS left out ****
                         085A  1413 ;
```

```
                0076   31  085A  1414          BRW     CTERM_QIODONE_EXIT      ; Exit back to common code
                           085D  1415
                           085D  1416 READ_CHAR_DONE:                         ; message type = 10
                           085D  1417
 00000000'EF    D7  085D  1418          DECL    READQIO                 ; Clear read active
                           0863  1419
  52    2C AB   9E  0863  1420          MOVAB   CTP$W_CH_PARAM(R11),R2  ; address of requested characteristics
        50      DD  0867  1421          PUSHL   R0                      ; Save R0
      F794'     30  0869  1422          BSBW    GETBUF                  ; Get buffer to write back to net
     53   50     D0  086C  1423          MOVL    R0,R3                   ; Set input
        50 8ED0   D0  086F  1424          POPL    R0                      ; Restore
                           0872  1425
     0FFD 8F    BB  0872  1426          PUSHR   #^M<R0,R2,R3,R4,R5, -
                           0876  1427                  R6,R7,R8,R9,R10,R11>    ; ??? this many???
  53    2C A3   9E  0876  1428          MOVAB   CTP$W_CH_PARAM(R3),R3   ; address to stuff CHAR message data
59 00000000'EF    9E  087A  1429          MOVAB   CHAR_BLOCK,R9           ; Characteristics block
  5A    04 A0   9E  0881  1430          MOVAB   AST$Q_IOSB(R0),R10      ; IOSB from sense mode
                           0885  1431
      F778'     30  0885  1432          BSBW    CTSENSECHAR             ; Returns R1 as length of fields
     0FFD 8F    BA  0888  1433          POPR    #^M<R0,R2,R3,R4,R5, -
                           088C  1434                  R6,R7,R8,R9,R10,R11>
                           088C  1435
     51    03    C0  088C  1436          ADDL    #3,R1                   ; Plus 3 for msgtype, flags
  26 A3    09    9B  088F  1437          MOVZBW  #PRO$C_DATA, -
                           0893  1438                  CTP$B_PRO_MSGTYPE(R3)   ; Set foundation data message
  2A A3    0B    9B  0893  1439          MOVZBW  #CTP$C_MT_CHAR, -
                           0897  1440                  CTP$B_MSGTYPE(R3)       ; Set message type
  52    26 A3   9E  0897  1441          MOVAB   CTP$B_PRO_MSGTYPE(R3),R2 ; Address of message
                           089B  1442
        0035    31  089B  1443          BRW     CTERM_QIODONE_EXIT      ; Exit back to common code
                           089E  1444
                           089E  1445 CHAR_DONE:                          ; message type = 11
                           089E  1446
 00000000'EF    D7  089E  1447          DECL    READQIO                 ; Clear read active
                           08A4  1448
        002C    31  08A4  1449          BRW     CTERM_QIODONE_EXIT      ; Exit back to common code
                           08A7  1450
                           08A7  1451 CHECK_INP_DONE:                     ; message type = 12
                           08A7  1452
        50      DD  08A7  1453          PUSHL   R0                      ; Save R0
      F754'     30  08A9  1454          BSBW    GETBUF                  ; Get buffer to write back to net
     53   50     D0  08AC  1455          MOVL    R0,R3                   ; Set input
        50 8ED0   D0  08AF  1456          POPL    R0                      ; Restore
                           08B2  1457
  26 A3    09    9B  08B2  1458          MOVZBW  #PRO$C_DATA, -
                           08B6  1459                  CTP$B_PRO_MSGTYPE(R3)   ; Set foundation data (cterm) message
  2A A3    0D    9B  08B6  1460          MOVZBW  #CTP$C_MT_INP_COUNT, -
                           08BA  1461                  CTP$B_MSGTYPE(R3)       ; Set message type
  52    26 A3   9E  08BA  1462          MOVAB   CTP$B_PRO_MSGTYPE(R3),R2 ; Address of message
     2C AB      D0  08BE  1463          MOVL    CTP$W_IC_COUNT(R11),-
     2C A3      08C1  1464                  CTP$W_IC_COUNT(R3)      ; Copy data (3 bytes, really)
     51    04    9A  08C3  1465          MOVZBL  #CTP$C_IC_PROLEN,R1     ; Set length
        04    E1  08C6  1466          BBC     #FLG$V_VAXHOST,-
02 00000000'EF      08C8  1467                  CTERM_FLAG,10$
        51    D6  08CE  1468          INCL    R1                      ; also copy character
                           08D0  1469
        0000    31  08D0  1470 10$:     BRW     CTERM_QIODONE_EXIT      ; Exit back to common code
```

```
                        08D3  1471
                        08D3  1472 CTERM_QIODONE_EXIT:
                        08D3  1473
                        08D3  1474 ;
                        08D3  1475 ; At this point, should be set up as follows:
                        08D3  1476 ;
                        08D3  1477 ; R0 - Same as input to CTERM_QIODONE (points to main data block)
                        08D3  1478 ; R1 - Length of message to write to net (minus four bytes for header) (if R3 > 0)
                        08D3  1479 ; R2 - Address of message to write to net (if R3 > 0)
                        08D3  1480 ; R3 - AST block for write to net (or 0)
                        08D3  1481 ;
                  0E    BB  08D3  1482          PUSHR   #^M<R1,R2,R3>           ; *** don't trust anybody
                  16    10  08D5  1483          BSBB    MESSAGE_END             ; start up another QIO?
                  0E    BA  08D7  1484          POPR    #^M<R1,R2,R3>           ;
                        08D9  1485
              50  53    D0  08D9  1486          MOVL    R3,R0                   ; Set return
                  0E    13  08DC  1487          BEQL    10$
           02 A2  51    B0  08DE  1488          MOVW    R1,-
                            08E2  1489                  CTP$W_MSGSIZE-CTP$B_PRO_MSGTYPE(R2) ; Set size
              51  04    C0  08E2  1490          ADDL    #4,R1                   ; Size to write to net
      60  00000000'EF  9E  08E5  1491          MOVAB   LNKWRTDONE,AST$L_STATE(R0)      ; Set ast routine (next state)
                        08EC  1492 10$:
                        05  08EC  1493          RSB                             ; Return to common path
                        08ED  1494
                        08ED  1495
                        08ED  1496 MESSAGE_END:
                        08ED  1497
           51   0E A0   3C  08ED  1498          MOVZWL  AST$W_OFFSET(R0),R1     ; Fetch current offset
        5B   51   50    C1  08F1  1499          ADDL3   R0,R1,R11               ; Add base + offset
     51    28 AB   02    A1  08F5  1500          ADDW3   #2,CTP$W_MSGSIZE(R11),R1 ; Add two bytes for message size
           0E A0   51    A0  08FA  1501          ADDW2   R1,AST$W_OFFSET(R0)     ; Add to offset
                        08FE  1502
              0E A0     B1  08FE  1503          CMPW    AST$W_OFFSET(R0),-
              10 A0         0901  1504                  AST$W_BUFSIZ(R0)        ; Compare to size read
                 06     18  0903  1505          BGEQ    MESSAGE_DONE            ; Branch if done
                        0905  1506
  F808 CF       00    FB  0905  1507          CALLS   #0,W^CTERM_PROCMSG      ; process message
                        05  090A  1508          RSB                             ; Return
                        090B  1509
                        090B  1510 MESSAGE_DONE:
                        090B  1511
                  0F    91  090B  1512          CMPB    #CTP$C_MT_VMSQIO,-
              2A A0         090D  1513                  CTP$B_MSGTYPE(R0)       ; Was it a VMS QIO?
                  0E    13  090F  1514          BEQL    10$                     ; branch if yes
                        0911  1515
                  07    91  0911  1516          CMPB    #CTP$C_MT_WRITE,-
              2A A0         0913  1517                  CTP$B_MSGTYPE(R0)       ; Was it a write? ***
                  08    13  0915  1518          BEQL    10$
       00000000'EF      D7  0917  1519          DECL    READQIO                 ; *** can this be cleaned up?
                  06    11  091D  1520          BRB     15$
                        091F  1521 10$:
       00000000'EF      D7  091F  1522          DECL    WRITEQIO
                        0925  1523 15$:
                F6D8'   30  0925  1524          BSBW    BUFFREE                 ; Free this buffer
                        0928  1525 ;
                        0928  1526 ; Must start up whole thing again for queued messages
                        0928  1527 ;
```

```
                                0928  1528          MOVAB   WRITEQIO,R1                   ; address
50  00000000'FF   0F  092F  1529          REMQUE  @WRITEQ,R0                    ; fetch attempt
                13  1C  0936  1530          BVC     20$                           ; Branch if not empty
                            0938  1531
51  00000000'EF   9E  0938  1532          MOVAB   READQIO,R1                    ; address
50  00000000'FF   0F  093F  1533          REMQUE  @READQ,R0                     ; try other queue
                03  1C  0946  1534          BVC     20$                           ; Branch if not empty
              0081  31  0948  1535          BRW     30$                           ; Exit
                            094B  1536 20$:
                61  D6  094B  1537          INCL    (R1)                          ; Interlock
        F7C0 CF   00  FB  094D  1538          CALLS   #0,W^CTERM_PROCMSG            ; process message
                            0952  1539 ;
                            0952  1540 ; Decide if net needs to be "turned back on"
                            0952  1541 ;
 00000172'EF    D7  0952  1542          DECL    QUEUED_BUFS                   ; one less entry in queue now
 00000171'EF    95  0958  1543          TSTB    READNETFLAG                   ; has reading from net stopped?
                6C  13  095E  1544          BEQL    30$                           ; branch if not
                            0960  1545
                01  D1  0960  1546          CMPL    #MIN_QUEUED_BUFS,-
 00000172'EF                0962  1547                  QUEUED_BUFS                   ; hit lower limit?
                63  19  0967  1548          BLSS    30$                           ; Branch if not yet, still much work to
                            0969  1549                                                ;  remaining in queues
                            0969  1550
 00000171'EF    94  0969  1551          CLRB    READNETFLAG                   ; Clear flag
                            096F  1552
                            096F  1553 .IF DF DFBUG
                            096F  1554          movab   neton,r0              ; *** Debug
                            096F  1555          bsbw    log_ascic             ; ** Debug
                            096F  1556 .ENDC
              F68E'  30  096F  1557          BSBW    GETBUF                        ; Get buffer
60  00000000'EF   9E  0972  1558          MOVAB   LINKRECV,AST$L_STATE(R0)              ; Set ast address
                            0979  1559
                            0979  1560          $QIO_S  CHAN = LINKCHAN -             ; Read link again
                            0979  1561                  FUNC = #IO$_READVBLK -
                            0979  1562                  IOSB = AST$Q_IOSB(R0) -
                            0979  1563                  ASTADR = ASTHANDLER -
                            0979  1564                  ASTPRM = R0 -
                            0979  1565                  P1 = AST$T_BUF(R0) -
                            0979  1566                  P2 = #MAXMSG
                            09A4  1567          ONERROR QUIT                          ; Quit if error occurs
                            09CC  1568
                            09CC  1569 30$:
                            09CC  1570          ;
                            09CC  1571          ; Following test is here because a burst write can still
                            09CC  1572          ; be in progress when a hangup occurs (e.g. LOGOUT/FULL)
                            09CC  1573          ;
        0000'CF    95  09CC  1574          TSTB    W^WAKEFLAG                    ; Should RTPAD be exiting?
              0B   13  09D0  1575          BEQL    40$                           ; Branch if not
                            09D2  1576          $WAKE_S                               ; Wake up main thread
                            09DD  1577 40$:
                05  09DD  1578          RSB                                   ; return
                            09DE  1579
                            09DE  1580
```

```
                        09DE  1582
                        09DE  1583          .SBTTL  CTERM_LNKWRTDONE - Write to link has completed
                        09DE  1584
                        09DE  1585 ;++
                        09DE  1586 ;
                        09DE  1587 ; FUNCTIONAL DESCRIPTION:
                        09DE  1588 ;
                        09DE  1589 ;
                        09DE  1590 ; CALLING SEQUENCE:
                        09DE  1591 ;       NONE
                        09DE  1592 ;
                        09DE  1593 ; INPUT PARAMETERS:
                        09DE  1594 ;       NONE
                        09DE  1595 ;
                        09DE  1596 ; IMPLICIT INPUTS:
                        09DE  1597 ;       NONE
                        09DE  1598 ;
                        09DE  1599 ; OUTPUT PARAMETERS:
                        09DE  1600 ;       NONE
                        09DE  1601 ;
                        09DE  1602 ; IMPLICIT OUTPUTS:
                        09DE  1603 ;       NONE
                        09DE  1604 ;
                        09DE  1605 ; COMPLETION CODES:
                        09DE  1606 ;       NONE
                        09DE  1607 ;
                        09DE  1608 ; SIDE EFFECTS:
                        09DE  1609 ;       NONE
                        09DE  1610 ;
                        09DE  1611 ;--
                        09DE  1612
                        09DE  1613 CTERM_LNKWRTDONE::
                        09DE  1614
F61F'  30               09DE  1615          BSBW    BUFFREE
                        09E1  1616
       05               09E1  1617          RSB
                        09E2  1618
```

```
                     09E2  1620
                     09E2  1621          .SBTTL  CTERM_OUTBANDAST - Handle out of band ast
                     09E2  1622
                     09E2  1623 ;++
                     09E2  1624 ;
                     09E2  1625 ; FUNCTIONAL DESCRIPTION:
                     09E2  1626 ;
                     09E2  1627 ; AST routine for all out of bands enabled. Writes to net message
                     09E2  1628 ; containing character typed.
                     09E2  1629 ;
                     09E2  1630 ; CALLING SEQUENCE:
                     09E2  1631 ;
                     09E2  1632 ;       CALLS, CALLG
                     09E2  1633 ;
                     09E2  1634 ; INPUT PARAMETERS:
                     09E2  1635 ;
                     09E2  1636 ;       4(AP) - Character typed to generate AST
                     09E2  1637 ;
                     09E2  1638 ; IMPLICIT INPUTS:
                     09E2  1639 ;       NONE
                     09E2  1640 ;
                     09E2  1641 ; OUTPUT PARAMETERS:
                     09E2  1642 ;       NONE
                     09E2  1643 ;
                     09E2  1644 ; IMPLICIT OUTPUTS:
                     09E2  1645 ;
                     09E2  1646 ;       QIO done to net.
                     09E2  1647 ;
                     09E2  1648 ; COMPLETION CODES:
                     09E2  1649 ;       NONE
                     09E2  1650 ;
                     09E2  1651 ; SIDE EFFECTS:
                     09E2  1652 ;       NONE
                     09E2  1653 ;
                     09E2  1654 ;--
                     09E2  1655
                000C 09E2  1656 .ENTRY  CTERM_OUTBANDAST, ^M<R2,R3>     ; Save registers
                     09E4  1657         ;
                     09E4  1658         ; Check for discard output
                     09E4  1659         ;
                  52 D4    09E4  1660         CLRL    R2                      ; no discard flag
51   00000107'EF   9E    09E6  1661         MOVAB   OUTBAND_SET,R1          ; Set oob block
     50    04 AC   9A    09ED  1662         MOVZBL  4(AP),R0                ; Fetch byte
  10 18 A1   50   E1    09F1  1663         BBC     R0,OOB_DISCARD(R1),10$ ; check bit
                     09F6  1664         ASSUME  CTP$M_DS_DISCARD EQ 1   ; assume
                  52 96    09F6  1665         INCB    R2                      ; Set to 1
                  02 88    09F8  1666         BISB    #FLG$M_CTRL_CY,-
     00000000'EF           09FA  1667                 CTERM_FLAG              ; Set flushing flag
         0651 8F   3C    09FF  1668         MOVZWL  #SS$_CONTROLC,-
         0176'CF         0A03  1669                 W^FLUSH_STATUS          ; Set ^C in status
                     0A06  1670 10$:
           F5F7'   30    0A06  1671         BSBW    GETBUF                  ; Fetch buffer
                     0A09  1672
              09   9B    0A09  1673         MOVZBW  #PRO$C_DATA,-
           26 A0         0A0B  1674                 CTP$B_PRO_MSGTYPE(R0)   ; Set protocol message type
              03   B0    0A0D  1675         MOVW    #CTP$C_OB_PROLEN,-
           28 A0         0A0F  1676                 CTP$W_MSGSIZE(R0)       ; Set message size
```

```
                04   90  0A11  1677          MOVB    #CTP$C_MT_OUT_BAND,-
              2A A0      0A13  1678                  CTP$B_MSGTYPE(R0)       ; Set message type
          2B A0   52   90  0A15  1679          MOVB    R2,CTP$B_OB_FLAGS(R0)   ; set discard state
      2C A0   04 AC    90  0A19  1680          MOVB    4(AP),CTP$B_OB_CHAR(R0) ; Character typed
                           0A1E  1681
           51   07    9A  0A1E  1682          MOVZBL  #CTP$C_OB_MSGLEN,R1     ; Message length
        52   26 A0    9E  0A21  1683          MOVAB   CTP$B_PRO_MSGTYPE(R0),R2 ; Address of data
               F5D8'   30  0A25  1684          BSBW    WRITE_TO_NETX           ; Write message to NET
                           0A28  1685
                        04  0A28  1686          RET
```

```
                                0A29  1688          .SBTTL  CTERM_CTRLO_AST - Handle ^O out of band ast
                                0A29  1689
                                0A29  1690 ;++
                                0A29  1691 ;
                                0A29  1692 ; FUNCTIONAL DESCRIPTION:
                                0A29  1693 ;
                                0A29  1694 ; AST routine for all ^O's typed.
                                0A29  1695 ;
                                0A29  1696 ; CALLING SEQUENCE:
                                0A29  1697 ;
                                0A29  1698 ;       CALLS, CALLG
                                0A29  1699 ;
                                0A29  1700 ; INPUT PARAMETERS:
                                0A29  1701 ;       NONE
                                0A29  1702 ;
                                0A29  1703 ; IMPLICIT INPUTS:
                                0A29  1704 ;       NONE
                                0A29  1705 ;
                                0A29  1706 ; OUTPUT PARAMETERS:
                                0A29  1707 ;       NONE
                                0A29  1708 ;
                                0A29  1709 ; IMPLICIT OUTPUTS:
                                0A29  1710 ;
                                0A29  1711 ;       QIO done to net.
                                0A29  1712 ;
                                0A29  1713 ; COMPLETION CODES:
                                0A29  1714 ;       NONE
                                0A29  1715 ;
                                0A29  1716 ; SIDE EFFECTS:
                                0A29  1717 ;       NONE
                                0A29  1718 ;
                                0A29  1719 ;--
                                0A29  1720
                        000C    0A29  1721 .ENTRY  CTERM_CTRLO_AST, ^M<R2,R3>      ; Save registers
                                0A2B  1722
                 F5D2'  30      0A2B  1723         BSBW    GETBUF                  ; Fetch buffer
                                0A2E  1724
                    09  9B      0A2E  1725         MOVZBW  #PRO$C_DATA,-
                26  A0          0A30  1726                 CTP$B_PRO_MSGTYPE(R0)   ; Set protocol message type
                    02  B0      0A32  1727         MOVW    #CTP$C_DS_PROLEN,-
                28  A0          0A34  1728                 CTP$W_MSGSIZE(R0)       ; Set message size
                    09  90      0A36  1729         MOVB    #CTP$C_MT_DIS_STATE,-
                2A  A0          0A38  1730                 CTP$B_MSGTYPE(R0)       ; Set message type
                2B  A0  94      0A3A  1731         CLRB    CTP$B_DS_FLAGS(R0)      ; Assume discard output off now
                    02  E0      0A3D  1732         BBS     #FLG$V_CTRL_O,-
03 00000000'EF          0A3F  1733                 CTERM_FLAG,10$          ; Branch if state was "on"
                        0A45  1734         ASSUME  CTP$M_DS_DISCARD EQ 1   ; assume
                2B  A0  96      0A45  1735         INCB    CTP$B_DS_FLAGS(R0)      ; Set to 1
                        0A48  1736 10$:
                    04  8C      0A48  1737         XORB2   #FLG$M_CTRL_O,-
   00000000'EF          0A4A  1738                 CTERM_FLAG              ; Flop bit
                        0A4F  1739
                51  06  9A      0A4F  1740         MOVZBL  #CTP$C_DS_MSGLEN,R1     ; Message length
             52 26  A0  9E      0A52  1741         MOVAB   CTP$B_PRO_MSGTYPE(R0),R2 ; Address of data
                 F5A7'  30      0A56  1742         BSBW    WRITE_TO_NETX           ; Write message to NET
                        0A59  1743
                        04      0A59  1744         RET
```

```
                OASA  1745
```

```
                     0A5A  1747          .SBTTL  CTERM_UNSDATMBX - Unsolicited data mailbox
                     0A5A  1748
                     0A5A  1749 ;++
                     0A5A  1750 ;
                     0A5A  1751 ; FUNCTIONAL DESCRIPTION:
                     0A5A  1752 ;
                     0A5A  1753 ;
                     0A5A  1754 ; CALLING SEQUENCE:
                     0A5A  1755 ;       Called as an AST routine
                     0A5A  1756 ;
                     0A5A  1757 ; INPUT PARAMETERS:
                     0A5A  1758 ;       NONE
                     0A5A  1759 ;
                     0A5A  1760 ; IMPLICIT INPUTS:
                     0A5A  1761 ;       UNSDAT
                     0A5A  1762 ;
                     0A5A  1763 ; OUTPUT PARAMETERS:
                     0A5A  1764 ;       NONE
                     0A5A  1765 ;
                     0A5A  1766 ; IMPLICIT OUTPUTS:
                     0A5A  1767 ;       NONE
                     0A5A  1768 ;
                     0A5A  1769 ; COMPLETION CODES:
                     0A5A  1770 ;       NONE
                     0A5A  1771 ;
                     0A5A  1772 ; SIDE EFFECTS:
                     0A5A  1773 ;       NONE
                     0A5A  1774 ;
                     0A5A  1775 ;--
                     0A5A  1776
                000C 0A5A  1777 .Entry  CTERM_UNSDATMBX, ^M<R2,R3>
                     0A5C  1778
50  00000000'EF   9E 0A5C  1779         MOVAB   UNSDAT,R0               ; Address of data block
    01   2D A0   B1 0A63  1780         CMPW    CTP$W_BR_MSGCODE(R0), -
                     0A67  1781                 #MSG$_TRMUNSOLIC        ; Unsolicited data?
            0C   12 0A67  1782         BNEQ    10$                     ; branch if no
        51   06   3C 0A69  1783         MOVZWL  #CTP$C_IS_MSGLEN,R1     ; Length
52  00000053'EF   9E 0A6C  1784         MOVAB   CT_UNSDAT_MSG,R2        ; message
            39   11 0A73  1785         BRB     90$                     ; Continue
                     0A75  1786 10$:
    06   2D A0   B1 0A75  1787         CMPW    CTP$W_BR_MSGCODE(R0), -
                     0A79  1788                 #MSG$_TRMHANGUP         ; Hangup?
            11   12 0A79  1789         BNEQ    20$                     ; branch if no
        51   29   3C 0A7B  1790         MOVZWL  #UNBIND_MSGLEN,R1       ; Length
52  000000B6'EF   9E 0A7E  1791         MOVAB   UNBIND_MSG,R2           ; message
            04   B0 0A85  1792         MOVW    #UNBIND$C_DISCONNECT,-
        00DD'CF      0A87  1793                 W^UNBIND_WHY            ; Set reason as terminal disconnect
            22   11 0A8A  1794         BRB     90$                     ; Continue
                     0A8C  1795 20$:
 0053 8F   2D A0   B1 0A8C  1796         CMPW    CTP$W_BR_MSGCODE(R0), -
                     0A92  1797                 #MSG$_TRMBRDCST         ; Broadcast to mailbox?
            2D   12 0A92  1798         BNEQ    110$                    ; nope, unknown message type, ignore it
            09   9B 0A94  1799         MOVZBW  #PRO$C_DATA,-
        26 A0        0A96  1800                 CTP$B_PRO_MSGTYPE(R0)   ; set size, zero flags
            10   90 0A98  1801         MOVB    #CTP$C_MT_VMS_BRDCST,-
        2A A0        0A9A  1802                 CTP$B_MSGTYPE(R0)       ; Set messsage type
        2B A0   B4 0A9C  1803         CLRW    CTP$W_BR_FLAGS(R0)      ; clear flags
```

```
                51   41 A0    3C   0A9F  1804          MOVZWL  CTP$W_BR_MSGLEN(R0),R1  ; broadcast message length
                   51   1D    C0   0AA3  1805          ADDL2   #CTP$T_BR_MSGTXT-CTP$B_PRO_MSGTYPE,R1 ; Length of net write
                        04    A3   0AA6  1806          SUBW3   #CTP$B_MSGTYPE-CTP$B_PRO_MSGTYPE,-
                28 A0   51         0AA8  1807                  R1,CTP$W_MSGSIZE(R0)    ; Size of this message less overhead
                   52   50    D0   0AAB  1808          MOVL    R0,R2                   ; address of buffer
                                   0AAE  1809          ;
                                   0AAE  1810          ; Send message to net
                                   0AAE  1811          ;
                                   0AAE  1812 90$:
                60    C6'AF    9E   0AAE  1813          MOVAB   B^CTERM_UNSMSGDONE, -
                                   0AB2  1814                  AST$L_STATE(R0)         ; set ast address
                52    26 A2    9E   0AB2  1815          MOVAB   CTP$B_PRO_MSGTYPE(R2),R2 ; message data
           53   00000000'EF    9E   0AB6  1816          MOVAB   ASTHANDLER,R3           ; AST dispatcher
                      F540'    30   0ABD  1817          BSBW    WRITE_TO_NET            ; write to net
                               04   0AC0  1818          RET
                                   0AC1  1819 110$:
                C6'AF    00    FB   0AC1  1820          CALLS   #0,B^CTERM_UNSMSGDONE   ; call ast routine manually
                               04   0AC5  1821          RET                             ; return
                                   0AC6  1822
```

```
                        0AC6  1824          .SBTTL  CTERM_UNSMSGDONE - Mailbox message done
                        0AC6  1825
                        0AC6  1826 ;++
                        0AC6  1827 ;
                        0AC6  1828 ; FUNCTIONAL DESCRIPTION:
                        0AC6  1829 ;
                        0AC6  1830 ;
                        0AC6  1831 ; CALLING SEQUENCE:
                        0AC6  1832 ;       NONE
                        0AC6  1833 ;
                        0AC6  1834 ; INPUT PARAMETERS:
                        0AC6  1835 ;       NONE
                        0AC6  1836 ;
                        0AC6  1837 ; IMPLICIT INPUTS:
                        0AC6  1838 ;       NONE
                        0AC6  1839 ;
                        0AC6  1840 ; OUTPUT PARAMETERS:
                        0AC6  1841 ;       NONE
                        0AC6  1842 ;
                        0AC6  1843 ; IMPLICIT OUTPUTS:
                        0AC6  1844 ;       NONE
                        0AC6  1845 ;
                        0AC6  1846 ; COMPLETION CODES:
                        0AC6  1847 ;       NONE
                        0AC6  1848 ;
                        0AC6  1849 ; SIDE EFFECTS:
                        0AC6  1850 ;       NONE
                        0AC6  1851 ;
                        0AC6  1852 ;--
                        0AC6  1853
                0000    0AC6  1854 CTERM_UNSMSGDONE:: .WORD        0
                        0AC8  1855
50  00000000'EF   9E    0AC8  1856          MOVAB   UNSDAT,R0                   ; Address of block
    60    88 AF   9E    0ACF  1857          MOVAB   CTERM_UNSDATMBX, -
                        0AD3  1858                  AST$L_STATE(R0)             ; Set ast address
                        0AD3  1859          $QIO_S  CHAN    = TERMMBXCHAN,-
                        0AD3  1860                  FUNC    = #IO$_READVBLK,-
                        0AD3  1861                  ASTADR = ASTHANDLER,-
                        0AD3  1862                  ASTPRM = R0,-
                        0AD3  1863                  P1 = CTP$W_BR_MSGCODE(R0),-
                        0AD3  1864                  P2 = #MAXMSG-CTP$W_BR_MSGCODE   ; size
                        0AFD  1865          ONERROR QUIT
                     04 0B25  1866          RET
```

```
                     0B26  1868          .SBTTL CTERM_CTRL_CY - Control C or Control Y
                     0B26  1869
                     0B26  1870 ;++
                     0B26  1871 ;
                     0B26  1872 ; FUNCTIONAL DESCRIPTION:
                     0B26  1873 ;
                     0B26  1874 ; Set up state so that internal queues are flushed in response to ^C or ^Y.
                     0B26  1875 ;
                     0B26  1876 ; CALLING SEQUENCE:
                     0B26  1877 ;
                     0B26  1878 ;       CALLS #0,CTERM_CTRL_CY
                     0B26  1879 ;
                     0B26  1880 ; INPUT PARAMETERS:
                     0B26  1881 ;
                     0B26  1882 ;       AP - pointer to Control ast block
                     0B26  1883 ;
                     0B26  1884 ; IMPLICIT INPUTS:
                     0B26  1885 ;       NONE
                     0B26  1886 ;
                     0B26  1887 ; OUTPUT PARAMETERS:
                     0B26  1888 ;       NONE
                     0B26  1889 ;
                     0B26  1890 ; IMPLICIT OUTPUTS:
                     0B26  1891 ;
                     0B26  1892 ;       CTERM_FLAG
                     0B26  1893 ;       FLUSH_STATUS
                     0B26  1894 ;
                     0B26  1895 ; COMPLETION CODES:
                     0B26  1896 ;       NONE
                     0B26  1897 ;
                     0B26  1898 ; SIDE EFFECTS:
                     0B26  1899 ;       NONE
                     0B26  1900 ;
                     0B26  1901 ;--
                     0B26  1902
                0000 0B26  1903 .Entry  CTERM_CTRL_CY, 0
                     0B28  1904
             02   88 0B28  1905         BISB    #FLG$M_CTRL_CY,-
 00000000'EF         0B2A  1906                 CTERM_FLAG              ; Set flushing flag
                     0B2F  1907         ;
                     0B2F  1908         ; See if ^C or ^Y
                     0B2F  1909         ;
 0100 8F    04 AC    B1 0B2F  1910         CMPW    4(AP),#IO$M_CTRLCAST    ; check
               0C    13 0B35  1911         BEQL    10$                     ; branch if ^C
                     0B37  1912
         0611 8F    3C 0B37  1913         MOVZWL  #SS$_CONTROLY,-
         0176'CF         0B3B  1914                 W^FLOSH_STATUS          ; Set ^Y
         50   19    9A 0B3E  1915         MOVZBL  #TTY$C_CTRLY,R0         ;
               0A    11 0B41  1916         BRB     20$                     ; skip
                     0B43  1917 10$:
         0651 8F    3C 0B43  1918         MOVZWL  #SS$_CONTROLC,-
         0176'CF         0B47  1919                 W^FLOSH_STATUS          ; Set ^C
         50   03    9A 0B4A  1920         MOVZBL  #TTY$C_CTRLC,R0         ;
                     0B4D  1921 20$:
51   00000107'EF    9E 0B4D  1922         MOVAB   OUTBAND_SET,R1          ; Set oob block
   00 1C A1    50   E5 0B54  1923         BBCC    R0,OOB_ECHO(R1),30$     ; clear bit     *** testing (ABORT)
                     0B59  1924 30$:
```

```
                51    00000127'EF   9E   0B59  1925          MOVAB   OUTBAND_NEW,R1          ; Set oob block
                   00 1C A1    50   E5   0B60  1926          BBCC    R0,OOB_ECHO(R1),40$     ; clear bit     *** testing (ABORT)
                                    0B65  1927 40$:
                               50   DD   0B65  1928          PUSHL   R0                      ; push character
                    FE76 CF    01   FB   0B67  1929          CALLS   #1,W^CTERM_OUTBANDAST
                                    04   0B6C  1930          RET
```

```
                     0B6D  1932          .SBTTL CHECK_MOREDATA - Check for more data in typeahead
                     0B6D  1933 ;++
                     0B6D  1934 ;
                     0B6D  1935 ; FUNCTIONAL DESCRIPTION:
                     0B6D  1936 ;
                     0B6D  1937 ; Sense state of typeahead buffer, used in non-VAX host case
                     0B6D  1938 ;
                     0B6D  1939 ; CALLING SEQUENCE:
                     0B6D  1940 ;
                     0B6D  1941 ;       CALLS #0,CHECK_MOREDATA
                     0B6D  1942 ;
                     0B6D  1943 ; OUTPUT:
                     0B6D  1944 ;       R0 = 1 if more data in typeahead
                     0B6D  1945 ;       R0 = 0 if not
                     0B6D  1946 ;
                     0B6D  1947 ; COMPLETION CODES:
                     0B6D  1948 ;       NONE
                     0B6D  1949 ;
                     0B6D  1950 ;--
                     0B6D  1951
           0004      0B6D  1952 CHECK_MOREDATA: .WORD   ^M<R2>
                     0B6F  1953
     52    7E   7E   0B6F  1954          MOVAQ   -(SP),R2                ; 8 byte buffer
                     0B72  1955
                     0B72  1956          $QIOW_S CHAN = READCHAN, -
                     0B72  1957                  EFN = #2,-              ; efn ?
                     0B72  1958                  FUNC = #IO$_SENSEMODE!IO$M_TYPEAHDCNT,-         ; function code
                     0B72  1959                  P1   = (R2)
  04 50    E9        0B93  1960          BLBC    R0,20$                  ; branch if error
     62    B5        0B96  1961          TSTW    (R2)                    ; test for non zero count
     02    12        0B98  1962          BNEQ    30$                     ; exit with lbs in r0 if data in typeahead
                     0B9A  1963 20$:
     50    D4        0B9A  1964          CLRL    R0                      ; clear "data in typeahead"
                     0B9C  1965 30$:
           04        0B9C  1966          RET                             ; return
```

```
                     0B9D  1966          .SBTTL UNBIND_RECEIVED - unbind message received, check it out
                     0B9D  1969 ;++
                     0B9D  1970 ;
                     0B9D  1971 ; FUNCTIONAL DESCRIPTION:
                     0B9D  1972 ;
                     0B9D  1973 ;       HOST has requested UNBIND, give user reasonable exit message
                     0B9D  1974 ;
                     0B9D  1975 ; CALLING SEQUENCE:
                     0B9D  1976 ;
                     0B9D  1977 ;       BSBW    UNBIND_RECEIVED
                     0B9D  1978 ;
                     0B9D  1979 ; INPUT:
                     0B9D  1980 ;       R0 - CTP packet
                     0B9D  1981 ;
                     0B9D  1982 ; COMPLETION CODES:
                     0B9D  1983 ;       NONE
                     0B9D  1984 ;
                     0B9D  1985 ;--
                     0B9D  1986
                     0B9D  1987 UNBIND_RECEIVED:
                     0B9D  1988
52   27 A0   B0      0B9D  1989          MOVW    CTP$B_PRO_FILL(R0),R2           ; fetch reason code
                     0BA1  1990
   52   03   B1      0BA1  1991          CMPW    #UNBIND$C_USER,R2               ; logout?
        25   12      0BA4  1992          BNEQ    10$                             ; branch if no
                     0BA6  1993          QUIT    #SS$_NORMAL                     ; yes, exit
                     0BCB  1994 10$:
   52   04   B1      0BCB  1995          CMPW    #UNBIND$C_DISCONNECT,R2         ; hangup
        29   12      0BCE  1996          BNEQ    20$                             ; branch if no
                     0BD0  1997          QUIT    #SS$_HANGUP                     ; yes, exit
                     0BF9  1998 20$:
   52   07   B1      0BF9  1999          CMPW    #UNBIND$C_PROTERR,R2            ; protocol error?
        29   12      0BFC  2000          BNEQ    30$                             ; branch if no
                     0BFE  2001          QUIT    #REM$_PROTERR                   ; yes, exit
                     0C27  2002 30$:
                     0C27  2003          QUIT    #SS$_LINKDISCON                 ; quit
             05      0C50  2004          RSB
                     0C51  2005
                     0C51  2006 .END
```

```
$$T1                            = 00000001
AST$L_ITMLST                    = 00000016
AST$L_ODATA                     = 00000012
AST$L_STATE                     = 00000000
AST$Q_IOSB                      = 00000004
AST$T_BUF                       = 00000026
AST$W_BUFSIZ                    = 00000010
AST$W_OFFSET                    = 0000000E
ASTHANDLER                        ********    X   03
BADFIELD_ERROR                    00000100 R      03
BUFFREE                           ********    X   03
CANCEL_READ                       0000040D R      03
CHAR_BLOCK                        ********    X   03
CHAR_DONE                         0000089E R      03
CHAR_MSG                          00000516 R      03
CHECK_INP_DONE                    000008A7 R      03
CHECK_INP_MSG                     00000659 R      03
CHECK_MOREDATA                    00000B6D R      03
CLR_INPUT_DONE                    0000082A R      03
CLR_INPUT_MSG                     000003FA R      03
CLR_INPUT_RCV                     000000E0 R      03
CNTRCFLAG                         ********    X   03
CNTRLCHAN                         ********    X   03
CNTRLC_AST                        ********    X   03
CTERM$LOG_IO                      ********    X   03
CTERM_CTRLO_AST                   00000A29 RG     03
CTERM_CTRL_CY                     00000B26 RG     03
CTERM_ECO                         0000017E R      02
CTERM_FLAG                        ********    X   03
CTERM_LINKRECV                    00000041 RG     03
CTERM_LINKRECV_EXIT               000000CF R      03
CTERM_LNKWRTDONE                  000009DE RG     03
CTERM_OUTBANDAST                  000009E2 RG     03
CTERM_PROCMSG                     00000112 R      03
CTERM_PROCMSG_EXIT                000006DF R      03
CTERM_PROCMSG_QIO                 00000665 R      03
CTERM_QIODONE                     000006F1 RG     03
CTERM_QIODONE_EXIT                000008D3 R      03
CTERM_UNSDATMBX                   00000A5A RG     03
CTERM_UNSMSGDONE                  00000AC6 RG     03
CTERM_VERSION                     0000017D R      02
CTERM_VMSQIO                      ********    X   03
CTP$B_CI_FLAGS                  = 0000002B
CTP$B_DS_FLAGS                  = 0000002B
CTP$B_IN_PARMTYPE               = 00000037
CTP$B_IN_VERSION                = 0000002C
CTP$B_MSGTYPE                   = 0000002A
CTP$B_OB_CHAR                   = 0000002C
CTP$B_OB_FLAGS                  = 0000002B
CTP$B_PRO_FILL                  = 00000027
CTP$B_PRO_MSGTYPE               = 00000026
CTP$B_RD_CURS_POS               = 0000002F
CTP$B_RD_FLAGS                  = 0000002B
CTP$B_WR_POSTFIX                = 0000002E
CTP$B_WR_PREFIX                 = 0000002D
CTP$C_DS_MSGLEN                 = 00000006
CTP$C_DS_PROLEN                 = 00000002
CTP$C_IC_PROLEN                 = 00000004
CTP$C_IS_MSGLEN                 = 00000006
CTP$C_MT_CHAR                   = 0000000B
CTP$C_MT_DIS_STATE              = 00000009
CTP$C_MT_INIT                   = 00000001
CTP$C_MT_INP_COUNT              = 0000000D
CTP$C_MT_INP_STATE              = 0000000E
CTP$C_MT_OUT_BAND               = 00000004
CTP$C_MT_READ_DATA              = 00000003
CTP$C_MT_START_RD               = 00000002
CTP$C_MT_VMSQIO                 = 0000000F
CTP$C_MT_VMS_BRDCST             = 00000010
CTP$C_MT_VMS_READVFY            = 00000011
CTP$C_MT_WRITE                  = 00000007
CTP$C_MT_WRITE_COM              = 00000008
CTP$C_OB_MSGLEN                 = 00000007
CTP$C_OB_PROLEN                 = 00000003
CTP$C_RD_PROLEN                 = 00000008
CTP$C_SR_ALLBUTX                = 00000003
CTP$C_SR_EDIT                   = 00000002
CTP$C_SR_NORMTERM               = 00000002
CTP$C_SR_PREVTERM               = 00000000
CTP$C_SR_THISTERM               = 00000001
CTP$C_WC_PROLEN                 = 00000006
CTP$L_SR_FLAGS                  = 0000002B
CTP$M_DS_DISCARD                = 00000001
CTP$M_IS_NONZERO                = 00000001
CTP$M_RD_ABSTOKEN               = 00000008
CTP$M_RD_INPFULL                = 00000004
CTP$M_RD_INVESC                 = 00000002
CTP$M_RD_NORMAL                 = 00000000
CTP$M_RD_OUTBAND                = 00000003
CTP$M_RD_OVERUN                 = 0000000D
CTP$M_RD_PARITY                 = 0000000C
CTP$M_RD_TIMEOUT                = 00000005
CTP$M_RD_UNREAD                 = 00000006
CTP$M_RD_VALESC                 = 00000001
CTP$M_WR_BEFAFTRE               = 00000003
CTP$M_WR_BEGIN                  = 00000010
CTP$M_WR_END                    = 00000020
CTP$S_SR_CONTROL                = 00000003
CTP$S_SR_TERM_SET               = 00000002
CTP$S_WR_LOCK                   = 00000002
CTP$T_BR_MSGTXT                 = 00000043
CTP$T_RD_DATA                   = 00000032
CTP$T_SR2_TERM                  = 00000042
CTP$T_SR_TERM                   = 0000003A
CTP$T_WR_DATA                   = 0000002F
CTP$V_RD_MOR_DATA               = 00000004
CTP$V_SR_CONTROL                = 00000008
CTP$V_SR_CVTLOW                 = 00000006
CTP$V_SR_ESCAPE                 = 00000011
CTP$V_SR_NOECHO                 = 0000000B
CTP$V_SR_NOEDIT                 = 00000012
CTP$V_SR_PURGE                  = 00000002
CTP$V_SR_TERM_SET               = 0000000E
CTP$V_SR_TIMED                  = 0000000D
```

```
CTP$V_SR_TRMECHO                = 0000000C
CTP$V_WR_BEGIN                  = 00000004
CTP$V_WR_DISCARD                = 00000003
CTP$V_WR_END                    = 00000005
CTP$V_WR_LOCK                   = 00000000
CTP$V_WR_NEWLINE                = 00000002
CTP$V_WR_POSTFIX                = 00000008
CTP$V_WR_PREFIX                 = 00000006
CTP$V_WR_STATUS                 = 0000000A
CTP$V_WR_TRANSPARENT            = 0000000B
CTP$W_BR_FLAGS                  = 0000002B
CTP$W_BR_MSGCODE                = 0000002D
CTP$W_BR_MSGLEN                 = 00000041
CTP$W_CH_PARAM                  = 0000002C
CTP$W_IC_COUNT                  = 0000002C
CTP$W_MSGSIZE                   = 00000028
CTP$W_RD_TERM_POS               = 00000030
CTP$W_SR2_ALTECHSIZE            = 0000003A
CTP$W_SR2_EDITFLAGS             = 0000003E
CTP$W_SR2_FILLCHAR              = 00000040
CTP$W_SR2_PICSTRSIZE            = 0000003C
CTP$W_SR_END_DATA               = 00000030
CTP$W_SR_END_PRMT               = 00000034
CTP$W_SR_MAX_LEN                = 0000002E
CTP$W_SR_STR_DISP               = 00000036
CTP$W_SR_TIMEOUT                = 00000032
CTP$W_WC_HORPOS                 = 0000002C
CTP$W_WR_FLAGS                  = 0000002B
CTSENSECHAR                       ********    X   03
CT_BIND_ACC_MSG                   0000007F RG     02
CT_BIND_MSGLEN                  = 00000011   G
CT_CHAR_MSG                       ********    X   03
CT_INIT_CHAR                      00000047 R      02
CT_INIT_CHAR_MSGLEN             = 0000000E
CT_INIT_MSG                       00000000 R      02
CT_INIT_MSGLEN                  = 0000001F
CT_UNSDAT_MSG                     00000053 RG     02
CT_VMSQIO_DONE                    ********    X   03
ERROR_RCV                         000000F6 R      03
FLG$M_CTRL_CY                   = 00000002
FLG$M_CTRL_O                    = 00000004
FLG$V_CTERM                     = 00000000
FLG$V_CTRL_CY                   = 00000001
FLG$V_CTRL_O                    = 00000002
FLG$V_LOGGING                   = 00000003
FLG$V_VAXHOST                   = 00000004
FLUSH_STATUS                      00000176 R      02
GETBUF                            ********    X   03
INDFLAG                           ********    X   03
INIT_CHAR_STRT                    00000045 R      02
INIT_DONE                         0000075A R      03
INIT_MSG                          00000165 R      03
INIT_RCV                          000000E0 R      03
IO$M_BREAKTHRU                  = 00000200
IO$M_CANCTRLO                   = 00000040
IO$M_CTRLCAST                   = 00000100
IO$M_CVTLOW                     = 00000100
IO$M_ESCAPE                     = 00004000
IO$M_EXTEND                     = 00008000
IO$M_FCODE                      = 0000003F
IO$M_INCLUDE                    = 00000800
IG$M_NEWLINE                    = 00000400
IO$M_NOECHO                     = 00000040
IO$M_NOFILTR                    = 00000200
IO$M_NOFORMAT                   = 00000100
IO$M_OUTBAND                    = 00000400
IO$M_PURGE                      = 00000800
IO$M_REFRESH                    = 00002000
IO$M_TIMED                      = 00000080
IO$M_TRMNOECHO                  = 00001000
IO$M_TT_ABORT                   = 00001000
IO$M_TYPEAHDCNT                 = 00000040
IO$V_TIMED                      = 00000007
IO$_READPROMPT                  = 00000037
IO$_READVBLK                    = 00000031
IO$_SENSEMODE                   = 00000027
IO$_SETMODE                     = 00000023
IO$_TTYREADPALL                 = 0000003B
IO$_WRITEVBLK                   = 00000030
LEGALMSG                          00000187 R      02
LIB$SIGNAL                        ********    X   03
LINKCHAN                          ********    X   03
LINKRECV                          ********    X   03
LNKWRTDONE                        ********    X   03
LOG_ADDR                          000005C1 R      02
LOG_BUF                           000001A7 R      02
MAXMSG                          = 0000041A
MAXREAD                           00000183 R      02
MAXSENDMSG                        0000017F R      02
MAX_QUEUED_BUFS                 = 00000003
MESSAGE_DONE                      0000090B R      03
MESSAGE_END                       000008ED R      03
MIN_QUEUED_BUFS                 = 00000001
MSG$_TRMBRDCST                  = 00000053
MSG$_TRMHANGUP                  = 00000006
MSG$_TRMUNSOLIC                 = 00000001
NOT_DATA                          00000000 R      03
OOB_ABORT                       = 00000014
OOB_DISCARD                     = 00000018
OOB_ECHO                        = 0000001C
OOB_EXCLUDE                     = 00000004
OOB_INCLUDE                     = 0000000C
OOB_LEN                         = 00000020
OUTBANDABO                        ********    X   03
OUTBANDEXC                        ********    X   03
OUTBANDINC                        ********    X   03
OUTBAND_NEW                       00000127 RG     02
OUTBAND_SET                       00000107 R      02
PRO$C_ACCEPT                    = 00000004
PRO$C_DATA                      = 00000009
PRO$C_UNBIND                    = 00000002
QIODONE                           ********    X   03
QUEUED_BUFS                       00000172 R      02
RD_STAT_END                       0000016F R      02
```

```
RD_STAT_TBL                     00000147 R      02
READCHAR                        ********   X    03
READNETFLAG                     00000171 R      02
READQ                           ********   X    03
READQIO                         ********   X    03
READ_CHAR_DONE                  0000085D R      03
READ_CHAR_MSG                   00000500 R      03
RECORD_QUIT                     ********   X    03
REM$_BADMSG                     ********   X    03
REM$_INTERR                     ********   X    03
REM$_PROTERR                    ********   X    03
REM$_QIOERR                     ********   X    03
REM$_UNKMSG                     ********   X    03
RETSTATUS                       ********   X    03
SAVE_WR_FLAGS                   0000017B R      02
SAVE_WR_POSTFIX                 0000017A R      02
SS$_ABORT                   =   0000002C
SS$_BADESCAPE               =   0000003C
SS$_CONTROLC                =   00000651
SS$_CONTROLY                =   00000611
SS$_DATAOVERUN              =   00000838
SS$_EXQUOTA                 =   0000001C
SS$_HANGUP                  =   000002CC
SS$_LINKDISCON              =   000020EC
SS$_NORMAL                  =   00000001
SS$_OPINCOMPL               =   000002D4
SS$_PARITY                  =   000001F4
SS$_PARTESCAPE              =   000001FC
SS$_TIMEOUT                 =   0000022C
START_READ_DONE                 0000075D R      03
START_READ_MSG                  00000210 R      03
SYS$CANCEL                      ********  GX    03
SYS$DCLAST                      ********  GX    03
SYS$QIO                         ********  GX    03
SYS$QIOW                        ********  GX    03
SYS$SETAST                      ********  GX    03
SYS$WAKE                        ********  GX    03
TERMMBXCHAN                     ********   X    03
TERMSET                         000000DF R      02
TRM$K_EM_RDVERIFY           =   00000001
TRM$V_TM_NOEDIT             =   0000000F
TRM$V_TM_R_JUST             =   00000011
TRM$_ALTECRSTR              =   00000009
TRM$_EDITMODE               =   00000001
TRM$_FILLCHR                =   00000007
TRM$_INIOFFSET              =   00000008
TRM$_INISTRNG               =   00000005
TRM$_MODIFIERS              =   00000000
TRM$_PICSTRNG               =   00000006
TRM$_PROMPT                 =   00000004
TRM$_TERM                   =   00000003
TRM$_TIMEOUT                =   00000002
TTY$C_CTRLC                 =   00000003
TTY$C_CTRLY                 =   00000019
TTY$C_ESCAPE                =   0000001B
UNBIND$C_DISCONNECT         =   00000004
UNBIND$C_PROTERR            =   00000007
UNBIND$C_USER               =   00000003
UNBIND_MSG                      000000B6 R      02
UNBIND_MSGLEN               =   00000029
UNBIND_RECEIVED                 00000B9D R      03
UNBIND_WHY                      000000DD R      02
UNREAD_DONE                     00000827 R      03
UNREAD_MSG                      000003F5 R      03
UNREAD_RCV                      000000E0 R      03
UNSDAT                          ********     X  03
VMSQIO_DONE                     00000754 R      03
VMSQIO_MSG                      00000159 R      03
VMS_INDREAD                     ********    X   03
WAKEFLAG                        ********    X   03
WRITECHAN                       ********    X   03
WRITEQ                          ********    X   03
WRITEQIO                        ********    X   03
WRITE_DONE                      0000082D R      03
WRITE_MSG                       00000449 R      03
WRITE_TO_NET                    ********    X   03
WRITE_TO_NETX                   ********    X   03
WRITE_TO_NET_SYNC               ********    X   03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RTPAD	000005C5 (1477.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
RTPAD	00000C51 (3153.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.06	00:00:01.46
Command processing	153	00:00:00.52	00:00:04.96
Pass 1	606	00:00:17.44	00:01:08.82
Symbol table sort	0	00:00:02.76	00:00:07.46
Pass 2	331	00:00:04.30	00:00:18.10
Symbol table output	36	00:00:00.19	00:00:00.91
Psect synopsis output	2	00:00:00.02	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1159	00:00:25.30	00:01:41.74

The working set limit was 2400 pages.
147280 bytes (288 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2404 non-local and 118 local symbols.
2006 source lines were read in Pass 1, producing 38 object records in Pass 2.
40 pages of virtual memory were used to define 39 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RTPAD.OBJ]RTPAD.MLB;1	6
_255DUA28:[SYS.OBJ]LIB.MLB;1	12
_255DUA28:[SYSLIB]STARLET.MLB;2	17
TOTALS (all libraries)	35

2541 GETS were required to define 35 macros.

There were no errors, warnings or information messages.

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

CTDSENSE
LIS

CTDVMSPEC
LIS

CTSENSERT
LIS

RSTSRT
LIS

CTERMRT
LIS

DTE_DF03
LIS

CTSETRT
LIS

CTDSET
LIS