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**CTDRIVER

```
  CCCCCCCC  TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
  CCCCCCCC  TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
CC              TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
CC              TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
CC              TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
CC              TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
CC              TT      DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
CC              TT      DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
CC              TT      DD      DD  RR  RR          II      VV      VV  EE          RR  RR
CC              TT      DD      DD  RR  RR          II      VV      VV  EE          RR  RR
CC              TT      DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
CC              TT      DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
  CCCCCCCC      TT      DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
  CCCCCCCC      TT      DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....

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  CTDRIVER - Command Terminal Protocol Driver
                     0000     2          .IDENT  'V04-000'
                     0000     3
                     0000     4 ;DEBUG = 1;      ; general debug on/off
                     0000     5 ;DEBUG_LOG = 1   ; logging on/off
                     0000     6
00000007             0000     7 flg$v_ctrlc = 7
00000080             0000     8 flg$m_ctrlc = 1@flg$v_ctrlc
                     0000     9 ;
                     0000    10 ;*****************************************************************************
                     0000    11 ;*                                                                           *
                     0000    12 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
                     0000    13 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
                     0000    14 ;*  ALL RIGHTS RESERVED.                                                     *
                     0000    15 ;*                                                                           *
                     0000    16 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
                     0000    17 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
                     0000    18 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
                     0000    19 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                     0000    20 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY        *
                     0000    21 ;*  TRANSFERRED.                                                             *
                     0000    22 ;*                                                                           *
                     0000    23 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE      *
                     0000    24 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
                     0000    25 ;*  CORPORATION.                                                             *
                     0000    26 ;*                                                                           *
                     0000    27 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
                     0000    28 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                     0000    29 ;*                                                                           *
                     0000    30 ;*                                                                           *
                     0000    31 ;*****************************************************************************
                     0000    32 ;
                     0000    33 ;++
                     0000    34 ;
                     0000    35 ; FACILITY:
                     0000    36 ;
                     0000    37 ;       VAX/VMS Remote Terminal Driver
                     0000    38 ;
                     0000    39 ; ABSTRACT:
                     0000    40 ;
                     0000    41 ;       This module contains the remote terminal driver routines.  This driver
                     0000    42 ;       is used by the application process side of the operation.  In other
                     0000    43 ;       words, it receives the QIO requests from the process that does not
                     0000    44 ;       have local access to the terminal.
                     0000    45 ;
                     0000    46 ;       This driver's primary function is to receive QIO system service
                     0000    47 ;       requests, repackage the QIO arguments, and hand the new package to
                     0000    48 ;       the tranport mechanism for delivery to the remote terminal
                     0000    49 ;       handler process on the system with local access to the terminal.
                     0000    50 ;       The transport mechanism is DECnet.  Netdriver is called directly
                     0000    51 ;       via the internal IRP mechanism.
                     0000    52 ;
                     0000    53 ; AUTHOR:
                     0000    54 ;
                     0000    55 ;       Jake VanNoy     3-Aug-1982
                     0000    56 ;
                     0000    57 ; MODIFICATION HISTORY:
```

```
0000     58 ;
0000     59 ;       V03-011 LMP0308         L. Mark Pilant,         31-Aug-1984  11:53
0000     60 ;               Change the default ACL state in to ORB to be a descriptor,
0000     61 ;               not a queue.
0000     62 ;
0000     63 ;       V03-010 JLV0388         Jake VanNoy             25-JUL-1984
0000     64 ;               Add CTERM bugcheck code.  Add code to do implicit
0000     65 ;               enable of ^C when set host from RSX.  Enhance
0000     66 ;               logic that returns SS$_CONTROLY to also return
0000     67 ;               SS$_CONTROLC (for BASIC group).  Add code to return
0000     68 ;               SS$_INCOMPL for read verify.
0000     69 ;
0000     70 ;       V03-009 JLV0359         Jake VanNoy             11-JUL-1984
0000     71 ;               Add paranoia code to IOPST.  Fix bug that returned
0000     72 ;               extened mode IOSB for regular read.  Pick up SYSPASSWORD
0000     73 ;               from HOST system, not server.
0000     74 ;
0000     75 ;       V03-008 JLV0349         Jake VanNoy             10-APR-1984
0000     76 ;               Send UNBIND at refc=0 and setmode HANGUP.
0000     77 ;               Fix bug in writes larger than single buffer.
0000     78 ;               Make sure UNREAD message is sent from CANCEL routine.
0000     79 ;               This fixes a problem exhibited by the PHONE utility.
0000     80 ;               Fix bug that did not set CANCEL ^O on first write
0000     81 ;               after DISCARD OFF received.  Fix all calls to COM$DELxxxAST
0000     82 ;               to have PID as input.
0000     83 ;
0000     84 ;       V03-007 LMP0221         L. Mark Pilant,         30-Mar-1984  11:43
0000     85 ;               Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
0000     86 ;               ORB$W_PROT.
0000     87 ;
0000     88 ;       V03-006 JLV0338         Jake VanNoy              1-MAR-1984
0000     89 ;               Replace PSECT removed in JLV0335 that set CT_END after
0000     90 ;               remainder of modules.
0000     91 ;
0000     92 ;       V03-005 JLV0335         Jake VanNoy             28-FEB-1984
0000     93 ;               Clean up SET MODE function code parsing. Add support
0000     94 ;               for set and sense broadcast, set pid.  Return SS$_DEVREQERR
0000     95 ;               for modem or maintenance functions.
0000     96 ;               Make check in WRITE FDT routine to be sure that
0000     97 ;               current PID equals PID doing last write.  If
0000     98 ;               they are not equal, an old style single shot
0000     99 ;               write is done.  Rewrite out of band handling.
0000    100 ;               Add use of cursor position offset from EOL in READ_DATA.
0000    101 ;               Shorten CTP Queue parameters. Removed CT_BROADCAST routine.
0000    102 ;               Remove debug code.  Remove use of CTP$L_IRP.
0000    103 ;
0000    104 ;       V03-004 JLV0310         Jake VanNoy             10-OCT-1983
0000    105 ;               Clean up logging assembly switches for PLUTO team.
0000    106 ;               Fix race condition in FDT_WRITE that causes system
0000    107 ;               crashes.  Change restriction in FDT_READ that blows
0000    108 ;               away reads too large for PLUTO.
0000    109 ;
0000    110 ;       V03-003 JLV0287         Jake VanNoy             28-JUL-1983
0000    111 ;               Numerous cleanup bits. Add INIT mesage parsing. Add VMS
0000    112 ;               specific INIT parameter #4. Change VMS specific QIO
0000    113 ;               protocol to use extended CTERM bits (specified in INIT).
0000    114 ;               Use receipt INIT to start LOGINOUT. Add checking of
```

```
0000   115 ;               message length received from net. Add sense typeahead
0000   116 ;               to CTERM (CHECK_INPUT and INPUT_COUNT messages).
0000   117 ;               Change characteristic selectors over to new format.
0000   118 ;               Use new $TTYUCBDEF for UCB symbols. Add IO$M_NEWLINE.
0000   119 ;               Add read verify interface.
0000   120 ;
0000   121 ;--
```

```
                     0000   123          $TSADEF                 ; Cterm definitions
                     0000   124
                     0000   125 .ENABLE SUPPRESSION
                     0000   126
                     0000   127          $ACBDEF                 ; AST control block
                     0000   128          $AQBDEF                 ; ACP queue block
                     0000   129          $CANDEF                 ; Cancel interface codes
                     0000   130          $CRBDEF                 ; Channel request block
                     0000   131          $DCDEF                  ; Device classes and types
                     0000   132          $DDBDEF                 ; Device data block
                     0000   133          $DEVDEF                 ; Device characteristics
                     0000   134          $DYNDEF                 ; Buffer type codes
                     0000   135          $IDBDEF                 ; Interrupt data block
                     0000   136          $IODEF                  ; I/O function codes
                     0000   137          $IPLDEF                 ; Hardware IPL definitions
                     0000   138          $IRPDEF                 ; I/O request packet
                     0000   139          $JIBDEF                 ; Job Information block
                     0000   140          $MSGDEF                 ; Mailbox message types
                     0000   141          $ORBDEF                 ; Object's Rights Block
                     0000   142          $PCBDEF                 ; Process control block
                     0000   143          $PRDEF                  ; Processor registers
                     0000   144          $PRVDEF                 ; Privilege bits
                     0000   145          $PSLDEF                 ; Processor status longword
                     0000   146 ;***      $REMDEF                 ; General constants
                     0000   147          $SSDEF                  ; System status codes
                     0000   148          $TASTDEF                ; Out of band AST blocks
                     0000   149          $TQEDEF                 ; Timer Queue Entry
                     0000   150          $TRMDEF                 ; Item list definitions
                     0000   151          $TTDEF                  ; Terminal definitions
                     0000   152          $TT2DEF                 ; More definitions
                     0000   153          $TTYDEF                 ; Terminal driver definitions *** remove wit
                     0000   154          $TTYSYMDEF              ; TTY symbols and constants
                     0000   155          $UCBDEF                 ; Unit control block
                     0000   156          $TTYUCBDEF              ; local/remote terminal UCB
                     0000   157          $VCBDEF                 ; Volume control block
                     0000   158          $VECDEF                 ; Interrupt vector block
                     0000   159
                     0000   160 ;
                     0000   161 ; Local symbols
                     0000   162 ;
                     0000   163
                     0000   164 ;
                     0000   165 ; Argument list (AP) offsets for device-dependent QIO parameters
                     0000   166 ;
                     0000   167
00000000             0000   168 P1       = 0                     ; First QIO parameter
00000004             0000   169 P2       = 4                     ; Second QIO parameter
00000008             0000   170 P3       = 8                     ; Third QIO parameter
0000000C             0000   171 P4       = 12                    ; Fourth QIO parameter
00000010             0000   172 P5       = 16                    ; Fifth QIO parameter
00000014             0000   173 P6       = 20                    ; Sixth QIO parameter
                     0000   174
```

```
                     0000   176
                     0000   177 ;
                     0000   178 ; Other constants
                     0000   179 ;
                     0000   180
00000008             0000   181 CT$K_FIPL = 8                                ; IPL to synchronize
0000014B             0000   182 CT$K_MAXCTP = ^X88+^X88+12+^X2F              ; Size of write CTP to allocate ??
00000003             0000   183 CT$K_CTPQLIM = 3                             ; Maximum number of queued CTP's
00000001             0000   184 CT$K_CTPLOLIM = 1                            ; Desired low number of queued CTP's
001E8480             0000   185 CT$K_TQE_DELTA = 2000000                     ; tqe wakeup delta (200 milliseconds)
                     0000   186
                     0000   187 REMOTE_1 = <TT$M_NOECHO!TT$M_ESCAPE!TT$M_HOSTSYNC!TT$M_TTSYNC!TT$M_LOWER!-
                     0000   188             TT$M_MECHTAB!TT$M_WRAP!TT$M_CRFILL!TT$M_LFFILL!-
                     0000   189             TT$M_SCOPE!TT$M_SCRIPT!-
FF28DFFA             0000   190             TT$M_HOLDSCREEN!TT$M_EIGHTBIT!TT$M_MECHFORM!TT$M_MODEM!TT$M_PAGE>
                     0000   191
00000002             0000   192 REMOTE_2 = <TT2$M_AUTOBAUD>
                     0000   193
                     0000   194 ;
                     0000   195 ; Definitions that follow the standard UCB fields
                     0000   196 ;
                     0000   197
                     0000   198         .SAVE_LOCAL_BLOCK
                     0000   199         .PSECT $ABS$,ABS
                     0000   200
000000C0             0000   201 UCB$L_CT_RIIRP = UCB$L_RTT_NETIRP            ; Read Internal IRP address
                     0000   202
                     0000   203     $VIELD TQE,0,<-                         ; Flags in FR3 definition
                     0000   204         <BSY,1,M>,-                         ; TQE is queued
                     0000   205         <DELETE,1,M>-                       ; terminal has hungup, delete TQE
                     0000   206         >
                     0000   207
00000110             0000   208 . = UCB$T_CT_DEBUG_FILL              ; *** temporary
                     0110   209
                     0110   210 ; up to 10 longwords allowed here
                     0110   211
00000111             0110   212 UCB$B_CT_CRFILL:          .BLKB              ; CR fill
00000112             0111   213 UCB$B_CT_LFFILL:          .BLKB              ; LF fill
00000114             0112   214 UCB$W_CT_SPEED:           .BLKW              ; speed
                     0114   215
00000118             0114   216 UCB$L_CNT_TQE:           .BLKL              ; TQE flush count *** performance hook
0000011C             0118   217 UCB$L_CNT_OVR:           .BLKL              ; CTP overflow flush count *** performance h
00000120             011C   218 UCB$L_CNT_FRC:           .BLKL              ; forced flush count *** performance hook
00000124             0120   219 UCB$L_CNT_ADDR:          .BLKL              ; address
00000128             0124   220 UCB$L_CT_PID:            .BLKL              ; *** add to TTYUCBDEF
0000012C             0128   221 UCB$L_CT_INCLUDE:        .BLKL              ; ***V
00000130             012C   222 UCB$L_CT_EXCLUDE:        .BLKL              ;
00000134             0130   223 UCB$L_CT_ABORT:          .BLKL              ;
                     0134   224
                     0134   225
00000136             0134   226 UCB$W_CT_PARITY:          .BLKW              ; parity
00000138             0136   227                           .BLKW              ; spare
                     0138   228
                     0000   229     .RESTORE
                     0000   230 ;
                     0000   231 ; Redefinitions of the irp fields
                     0000   232 ;
```

```
                0000   233
00000040        0000   234 IRP$B_CT_RESPTYPE == IRP$Q_TT_STATE     ; Response type
00000041        0000   235 IRP$B_CT_CANCEL   = IRP$Q_TT_STATE + 1  ; Cancel flag
00000042        0000   236 IRP$W_CT_POST     = IRP$Q_TT_STATE + 2  ; I/O posted tag
                0000   237
```

```
                        .SBTTL  Standard tables

                ;
                ; Driver prologue table
                ;

                DPTAB   -                                       ; DPT-creation macro
                        END=CT_END,-                            ; End of driver label
                        ADAPTER=NULL,-                          ; Adapter type
                        UCBSIZE=<UCB$K_RTT_LENGTH>,-            ; Length of UCB
                        NAME=CTDRIVER                           ; Driver name

                DPT_STORE INIT                                  ; Start of load
                                                                ; initialization table
                DPT_STORE DDB,DDB$L_ACPD,L,<^A\REM\>            ; Default ACP name
                DPT_STORE DDB,DDB$L_ACPD+3,B,3                  ; ACP class
                DPT_STORE UCB,UCB$B_FIPL,B,CT$K_FIPL            ; Device fork IPL
                DPT_STORE UCB,UCB$B_DIPL,B,CT$K_FIPL            ; Device interrupt IPL
                DPT_STORE UCB,UCB$L_DEVCHAR,L,<-                ; Device characteristics
                        DEV$M_REC!-                             ;   record device
                        DEV$M_AVL!-                             ;   available
                        DEV$M_IDV!-                             ;   input device
                        DEV$M_ODV!-                             ;   output device
                        DEV$M_TRM!-                             ;   terminal device
                        DEV$M_CCL>                              ;   carriage control device
                DPT_STORE UCB,UCB$L_DEVCHAR2,L,<-               ; Device characteristics (2)
                        DEV$M_RTT>                              ; remote terminal UCB extension
                DPT_STORE UCB,UCB$W_STS,W,<-                    ; Device status
                        UCB$M_VALID>                            ; set valid
                DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_TERM         ; Terminal device
                DPT_STORE UCB,UCB$B_DEVTYPE,B,TT$_UNKNOWN       ; Unknown type
                DPT_STORE UCB,UCB$W_DEVBUFSIZ,aw,TTY$GW_DEFBUF  ; Default buffer size
                DPT_STORE UCB,UCB$L_DEVDEPEND,aL,TTY$GL_DEFCHAR ; Default characteristics
                DPT_STORE ORB,ORB$B_FLAGS,B,-                   ; Protection block flags
                                <ORB$M_PROT_16>                 ; SOGW protection word
                DPT_STORE ORB,ORB$W_PROT,aW,TTY$GW_PROT         ; Default allocation protection
                DPT_STORE ORB,ORB$L_OWNER,aL,TTY$GL_OWNUIC      ; Default owner UIC

                DPT_STORE REINIT                                ; Start of reload
                                                                ; initialization table
                DPT_STORE DDB,DDB$L_DDT,D,CT$DDT                ; Address of DDT
                DPT_STORE CRB,CRB$L_INTD+4,D,-                  ; Address of interrupt
                        CT_INTERRUPT                            ; service routine

                DPT_STORE END                                   ; End of initialization
                                                                ; tables
                ;
                ; Driver dispatch table
                ;
                DDTAB   -                                       ; DDT-creation macro
                        DEVNAM=CT,-                             ; Name of device
                        FUNCTB=CT_FUNCTABLE,-                   ; FDT address
                        UNSOLIC=CT_UNSOLIC,-                    ; Unsolicited attention routine
                        CANCEL=CT_CANCEL                        ; Cancel I/O routine

                ;
                ; Function dispatch table
```

```
0038    296 ;
0038    297
0038    298 CT_FUNCTABLE:                                       ; FDT for driver
0038    299         FUNCTAB ,-                                  ; Valid I/O functions
0038    300                 <READVBLK,-                         ; Read virtual
0038    301                 READLBLK,-                          ; Read logical
0038    302                 READPBLK,-                          ; Read physical
0038    303                 READPROMPT,-                        ; Read with prompt
0038    304                 TTYREADALL,-                        ; Read passall
0038    305                 TTYREADPALL,-                       ; Read with prompt passall
0038    306                 WRITEVBLK,-                         ; Write virtual
0038    307                 WRITELBLK,-                         ; Write logical
0038    308                 WRITEPBLK,-                         ; Write physical
0038    309                 SENSEMODE,-                         ; Sense device mode
0038    310                 SENSECHAR,-                         ; Sense device characteristics
0038    311                 SETMODE,-                           ; Set device mode
0038    312                 SETCHAR>                            ; Set device characteristics
0040    313         FUNCTAB ,-                                  ; Buffered functions
0040    314                 <READVBLK,-                         ; Read virtual
0040    315                 READLBLK,-                          ; Read logical
0040    316                 READPBLK,-                          ; Read physical
0040    317                 READPROMPT,-                        ; Read with prompt
0040    318                 TTYREADALL,-                        ; Read passall
0040    319                 TTYREADPALL,-                       ; Read with prompt passall
0040    320                 WRITEVBLK,-                         ; Write virtual
0040    321                 WRITELBLK,-                         ; Write logical
0040    322                 WRITEPBLK,-                         ; Write physical
0040    323                 SENSEMODE,-                         ; Sense device mode
0040    324                 SENSECHAR,-                         ; Sense device characteristics
0040    325                 SETMODE,-                           ; Set device mode
0040    326                 SETCHAR>                            ; Set device characteristics
0048    327         FUNCTAB CT_READ,-                           ; FDT read routine for
0048    328                 <READVBLK,-                         ; read virtual,
0048    329                 READLBLK,-                          ; read logical,
0048    330                 READPBLK,-                          ; read physical,
0048    331                 READPROMPT,-                        ; read with prompt
0048    332                 TTYREADALL,-                        ; read passall,
0048    333                 TTYREADPALL>                        ; and read with prompt passall
0054    334         FUNCTAB CT_WRITE,-                          ; FDT write routine for
0054    335                 <WRITEVBLK,-                        ; write virtual,
0054    336                 WRITELBLK,-                         ; write logical,
0054    337                 WRITEPBLK>                          ; and write physical.
0060    338         FUNCTAB CT_SENSEMODE,-                      ; FDT sense mode routine
0060    339                 <SENSECHAR,-                        ; for sense characteristics
0060    340                 SENSEMODE>                          ; and sense mode.
006C    341         FUNCTAB CT_SETMODE,-                        ; FDT set mode routine
006C    342                 <SETCHAR,-                          ; for set characteristics and
006C    343                 SETMODE>                            ; set mode.
```

```
                     0078   345 .SBTTL  Read status table
                     0078   346
                     0078   347 stat_table:
                0001 0078   348         .word   SS$_NORMAL              ; 0     terminator character
                0001 007A   349         .word   SS$_NORMAL              ; 1     valid escape sequence
                0001 007C   350         .word   SS$_NORMAL              ; 2     invalid escape sequence
                0611 007E   351         .word   SS$_CONTROLY            ; 3     out-of-band character (^C/^Y)
                0001 0080   352         .word   SS$_NORMAL              ; 4     input buffer full
                022C 0082   353         .word   SS$_TIMEOUT             ; 5     time out
                002C 0084   354         .word   SS$_ABORT               ; 6     unread  (*** why not CANCEL?)
                0001 0086   355         .word   SS$_NORMAL              ; 7     underflow
                01FC 0088   356         .word   SS$_PARTESCAPE          ; 8     absentee token
                0001 008A   357         .word   SS$_NORMAL              ; 9     vertical position change
                01F4 008C   358         .word   SS$_PARITY              ; 10    line break
                01F4 008E   359         .word   SS$_PARITY              ; 11    framing error
                01F4 0090   360         .word   SS$_PARITY              ; 12    parity error
                0838 0092   361         .word   SS$_DATAOVERUN          ; 13    receiver overrun
                02D4 0094   362         .word   SS$_OPINCOMPL           ; 14    VMS ONLY - operation incomplete
                0001 0096   363         .word   SS$_NORMAL              ; 15    15 is just for grins
                     0098   364
                     0098   365 .SBTTL  INIT MESSAGE BLOCK
                     0098   366
                     0098   367 .ENABL  LSB
                     0098   368
                     0098   369 INIT_MSGBLK:
                     0098   370
              0017' 0098   371 1$:     .WORD   20$-10$                 ; Length of cterm protocol
                0001 009A   372 10$:    .WORD   CTP$C_MT_INIT           ; init message type, zero flags
                  01 009C   373         .BYTE   1                       ; version 1
               00 03 009D   374         .BYTE   3,0                     ; 0 eco, customer mod
00000000 00000000 009F   375         .QUAD   0                       ; No revision
                  01 00A7   376         .BYTE   1                       ; parameter 1
                  02 00A8   377         .BYTE   2                       ; length in bytes
            00000039 00A9   378 W_INIT_BUFSIZ = .-1$+CTP$W_MSGSIZE      ; Offset to next field
                0000 00A9   379         .WORD   0                       ; data filled in
                     00AB   380
                  03 00AB   381         .BYTE   3                       ; parameter 3
                  04 00AC   382         .BYTE   4                       ; length in bytes
            0003FFFE 00AD   383         .LONG   ^B01111111111111111110  ; valid messages (1-17)
                     00B1   384 20$:
                     00B1   385         ;
                     00B1   386         ; end of INIT, also send initial characteristics here
                     00B1   387         ;
                     00B1   388         ; input state message here
                     00B1   389         ;
                0006 00B1   390         .WORD   CTP$C_CH_PROLEN+2       ; protocol length
                000B 00B3   391         .WORD   CTP$C_MT_CHAR           ; characteristics and flags
                0208 00B5   392         .WORD   <CH$C_CTERM@8!-
                     00B7   393                 CH$C_CT_INPUT_COUNT>    ; set input-count-state
                0002 00B7   394         .WORD   2                       ; no-read send
                     00B9   395
            00000021 00B9   396 C_INIT_MSGLEN = .-INIT_MSGBLK
                     00B9   397
                     00B9   398 .DSABL  LSB
```

```
                     00B9    400
00000002             00B9    401 POST_COUNT = 2          ; 0 is FINISHIO, 1 is ABORT
                     00B9    402
                     00B9    403 .MACRO  POST_IO,?L1
                     00B9    404
                     00B9    405         .LIST   MEB
                     00B9    406         MOVZBL  #POST_COUNT,R0          ; set position
                     00B9    407         BSBW    CHECK_POST_IO           ; check for error, set posted
                     00B9    408         JSB     G^COM$POST              ; post I/O
                     00B9    409         .NLIST  MEB
                     00B9    410
                     00B9    411 POST_COUNT = POST_COUNT + 1
                     00B9    412
                     00B9    413 .ENDM   POST_IO
                     00B9    414
                     00B9    415 .if df DEBUG_LOG
                     00B9    416
                     00B9    417 ;*** OPT macros are debug only
                     00B9    418
                     00B9    419 OPT$V_LOGGING = 0
                     00B9    420
                     00B9    421 .macro  IFOPT   option,label
                     00B9    422         .IF DF DEBUG_LOG
                     00B9    423         BBS     #OPT$V_'option',G^SGN$GL_VMSD4,'label'
                     00B9    424         .IFF
                     00B9    425         BBS     #OPT$V_'option',VMSD4,'label'
                     00B9    426         .ENDC
                     00B9    427 .ENDM   IFOPT
                     00B9    428
                     00B9    429 .macro  IFNOTOPT option,label
                     00B9    430         .IF DF DEBUG_LOG
                     00B9    431         BBC     #OPT$V_'option',G^SGN$GL_VMSD4,'label'
                     00B9    432         .IFF
                     00B9    433         BBC     #OPT$V_'option',VMSD4,'label'
                     00B9    434         .ENDC
                     00B9    435 .ENDM   IFNOTOPT
                     00B9    436
                     00B9    437 .endc           ; DEBUG_LOG
                     00B9    438
                     00B9    439
```

```
                     00B9   441          .SBTTL  CT_WRITE - Function Decision Routine for WRITE Functions
                     00B9   442 ;++
                     00B9   443 ; CT_WRITE - Function Decision Routine for WRITE Functions
                     00B9   444 ;
                     00B9   445 ; Functional description:
                     00B9   446 ;
                     00B9   447 ;       This routine is called by the SYS$QIO service to dispatch a WRITE
                     00B9   448 ;       I/O request.
                     00B9   449 ;
                     00B9   450 ;       The QIO parameters for terminal WRITES are:
                     00B9   451 ;
                     00B9   452 ;       P1 = address of the buffer
                     00B9   453 ;       P2 = size of the buffer
                     00B9   454 ;       P3 = (unused)
                     00B9   455 ;       P4 = carriage control specifier
                     00B9   456 ;
                     00B9   457 ;       The buffer is validated for access, the process's quota checked and
                     00B9   458 ;       decremented, the data and carriage control are copied to a message
                     00B9   459 ;       block, and the appropriate queueing of the message is done.
                     00B9   460 ;
                     00B9   461 ; Inputs:
                     00B9   462 ;
                     00B9   463 ;       R0-R2 = scratch registers
                     00B9   464 ;       R3 = address of the IRP (I/O request packet)
                     00B9   465 ;       R4 = address of the PCB (process control block)
                     00B9   466 ;       R5 = address of the UCB (unit control block)
                     00B9   467 ;       R6 = address of the CCB (channel control block)
                     00B9   468 ;       R7 = bit number of the I/O function code
                     00B9   469 ;       R8 = address of the FDT table entry for this routine
                     00B9   470 ;       R9-R11 = scratch registers
                     00B9   471 ;       AP = address of the 1st function dependent QIO parameter
                     00B9   472 ;
                     00B9   473 ; Outputs:
                     00B9   474 ;
                     00B9   475 ;       IRP$L_SVAPTE(R3) = address of message buffer
                     00B9   476 ;       IRP$W_BOFF(R3) = size of message buffer
                     00B9   477 ;       IRP$W_BCNT(R3) = size of user buffer (by EXE$WRITECHK)
                     00B9   478 ;
                     00B9   479 ;       The routine preserves all registers except R0-R2, and
                     00B9   480 ;       R6-R11.
                     00B9   481 ;
                     00B9   482 ;--
                     00B9   483
                     00B9   484 CTRLO:
           06   E0   00B9   485          BBS     #IO$V_CANCTRLO,-
       11 20 A3         00BB   486                  IRP$W_FUNC(R3),CAN_CTRLO ; If cancel ^O requested, do the write
  50    0609 8F   3C   00BE   487          MOVZWL  #SS$_CONTROLO,R0         ; Set status
        0852   31   00C3   488          BRW     FDT_FINISHIOC            ; Complete I/O
                     00C6   489
                     00C6   490 CT_WRITE:                                 ; WRITE FDT routine
                     00C6   491
        42 A3   B4   00C6   492          CLRW    IRP$W_CT_POST(R3)        ; Clear POSTed flag
           01   E0   00C9   493          BBS     #FLG$V_CTRLO,-
    EA 00DC C5         00CB   494                  UCB$W_CT_FLAGS(R5),CTRLO ; Branch if in CTRL O state
                     00CF   495 CAN_CTRLO:
                     00CF   496
           02   AA   00CF   497          BICW    #FLG$M_CTRLO,-
```

```
                00DC C5          00D1   498                     UCB$W_CT_FLAGS(R5)      ; Set no control O
                                 00D4   499             ;
                                 00D4   500             ; Process all inputs completely before determining where to
                                 00D4   501             ; copy write data. Start by probing the write data.
                                 00D4   502             ;
                   5A   57   D0  00D4   503             MOVL    R7,R10                  ; Save I/O function code number
                   56   6C   D0  00D7   504             MOVL    P1(AP),R6               ; Get user buffer virtual address
                   50   56   D0  00DA   505             MOVL    R6,R0                   ; Set up for write check call
                57   04   AC   3C  00DD   506             MOVZWL  P2(AP),R7               ; Get buffer size
                   51   57   D0  00E1   507             MOVL    R7,R1                   ; Set up for write check call
                        06   13  00E4   508             BEQL    10$                     ; Skip check if zero
            00000000'GF      16  00E6   509             JSB     G^EXE$WRITECHK          ; Check buffer access
                                 00EC   510                                             ; (no return means no access)
                                 00EC   511 10$:
                                 00EC   512             ;
                                 00EC   513             ; Map VMS QIO function modifiers into TSA function modifiers.
                                 00EC   514             ; Start by setting up CTP$W_WR_FLAGS in R9.
                                 00EC   515             ;
                     30 A3   B4  00EC   516             CLRW    IRP$W_BOFF(R3)          ; Make sure BOFF is zero
                   59   12   3C  00EF   517             MOVZWL  #<CTP$M_WR_BEGIN!-      ; Assume no status requested, no passall
                                 00F2   518                       CTP$M_WR_BEFAFT>,R9   ; no canctrlo and no refresh
                                 00F2   519                                             ; no prefix or postfix data, begin msg
                   5A   0B   91  00F2   520             CMPB    #IO$_WRITEPBLK,R10      ; Write physical?
                        05   12  00F5   521             BNEQ    20$                     ; Branch if not
               59   0800 8F   A8  00F7   522             BISW    #CTP$M_WR_TRANSPARENT,R9 ; Yes, set write transparent
                                 00FC   523 20$:
                 51   20 A3   3C  00FC   524             MOVZWL  IRP$W_FUNC(R3),R1       ; Function code
                        02   E4  0100   525             BBSC    #FLG$V_CANCTRLO,-
                0B 00DC C5       0102   526                     UCB$W_CT_FLAGS(R5),30$  ; Internal cancel ^O set (set by ^Y)
               51   2540 8F   B3  0106   527             BITW    #<IO$M_CANCTRLO!-
                                 010B   528                       IO$M_NOFORMAT!-
                                 010B   529                       IO$M_NEWLINE!-
                                 010B   530                       IO$M_REFRESH>,R1      ; Optimization, see if all options zero
                        1E   13  010B   531             BEQL    90$
                                 010D   532
                03 51   06   E1  010D   533             BBC     #IO$V_CANCTRLO,R1,40$   ; Cancel control O?
                   59   08   A8  0111   534 30$:       BISW    #CTP$M_WR_DISCARD,R9    ; Yes, set "do not discard" state
                                 0114   535 40$:
                05 51   08   E1  0114   536             BBC     #IO$V_NOFORMAT,R1,50$   ; Write passall?
               59   0800 8F   A8  0118   537             BISW    #CTP$M_WR_TRANSPARENT,R9 ; Yes, set write transparent
                                 011D   538 50$:
                03 51   0D   E1  011D   539             BBC     #IO$V_REFRESH,R1,60$    ; Refresh after write?
                   59   03   A8  0121   540             BISW    #CTP$M_WR_BEFAFTRE,R9   ; Yes, set refresh after unlock
                                 0124   541 60$:
                03 51   0A   E1  0124   542             BBC     #IO$V_NEWLINE,R1,90$    ; new line flag?
                   59   04   A8  0128   543             BISW    #CTP$M_WR_NEWLINE,R9    ; set bit (VMS specific)
                                 012B   544 90$:
                                 012B   545             ;
                                 012B   546             ; Set up carriage control
                                 012B   547             ;
                        5A   D4  012B   548             CLRL    R10                     ; use for prefix/postfix
            3C A3   0C   AC   D0  012D   549             MOVL    P4(AP),IRP$B_CARCON(R3) ; get carriage control
                        4E   13  0132   550             BEQL    140$                    ; ok to continue if none
              00000000'GF      16  0134   551             JSB     G^EXE$CARRIAGE          ; Call exec routine to map
                                 013A   552             ;
                                 013A   553             ; Set up prefix
                                 013A   554             ;
```

```
                51    3D A3    9A  013A    555          MOVZBL  IRP$B_CARCON+1(R3),R1   ; Fetch character field for prefix
                           11    13  013E    556          BEQL    100$                     ; If zero, then use newline count
                      51    0A    91  0140    557          CMPB    #TTY$C_LF,R1             ; Also if linefeed
                            0C    13  0143    558          BEQL    100$                     ;  use newline count
                        3C A3     95  0145    559          TSTB    IRP$B_CARCON(R3)         ; See if non-zero character count
                           13    13  0148    560          BEQL    115$                     ; Branch if zero
              59    0080 8F     A8  014A    561          BISW    #<CTP$C_WR_CHAR@ -
                                    014F    562                  CTP$V_WR_PREFIX>,R9      ; Set character flag
                           09    11  014F    563          BRB     110$                     ; continue
              59    0040 8F     A8  0151    564 100$:    BISW    #<CTP$C_WR_NEWLINECNT@ -
                                    0156    565                  CTP$V_WR_PREFIX>,R9      ; Set newline count flag
                51    3C A3     90  0156    566          MOVB    IRP$B_CARCON(R3),R1      ; Fetch newline count
                                    015A    567 110$:
                   5A    51     90  015A    568          MOVB    R1,R10                   ; Set newline count or character
                                    015D    569          ;
                                    015D    570          ; Set up postfix
                                    015D    571          ;
                                    015D    572 115$:
                51    3F A3     9A  015D    573          MOVZBL  IRP$B_CARCON+3(R3),R1    ; Fetch character field for postfix
                           11    13  0161    574          BEQL    120$                     ; If zero, then use newline count???
                      51    0A    91  0163    575          CMPB    #TTY$C_LF,R1             ; Also if linefeed
                            0C    13  0166    576          BEQL    120$                     ;  use newline count
                        3E A3     95  0168    577          TSTB    IRP$B_CARCON+2(R3)       ; See if non-zero character count
                           15    13  016B    578          BEQL    140$                     ; Branch if zero
              59    0200 8F     A8  016D    579          BISW    #<CTP$C_WR_CHAR@ -
                                    0172    580                  CTP$V_WR_POSTFIX>,R9     ; Set character flag
                           09    11  0172    581          BRB     130$                     ; continue
              59    0100 8F     A8  0174    582 120$:    BISW    #<CTP$C_WR_NEWLINECNT@ -
                                    0179    583                  CTP$V_WR_POSTFIX>,R9     ; Set newline count flag
                51    3E A3     90  0179    584          MOVB    IRP$B_CARCON+2(R3),R1    ; Fetch newline count
                                    017D    585 130$:
         5A   08    08    51     F0  017D    586          INSV    R1,#8,#8,R10             ; Set newline count or character
                                    0182    587 140$:
                                    0182    588          ;
                                    0182    589          ; At this point, all inputs have been verified. The remainder
                                    0182    590          ; of the work here determines where to set up the protocol
                                    0182    591          ; message. If buffering is off, a buffer is allocated and
                                    0182    592          ; queued immediately to the net. If buffering is on, a current
                                    0182    593          ; buffer is used, if possible, otherwise a new buffer is used.
                                    0182    594          ; For large writes, it is possible for multiple protocol messages
                                    0182    595          ; to be written.
                                    0182    596          ;
                     06    E0     0182    597          BBS     #FLG$V_BUFFER,-
                03 00DC C5          0184    598                  UCB$W_CT_FLAGS(R5),170$ ; Branch if not buffering
                                    0188    599 150$:
                      00C1    31    0188    600          BRW     400$                     ; handle non-buffered data
                                    018B    601          ;
                                    018B    602          ; Note that two processes can be in the write FDT routine for the
                                    018B    603          ; same terminal at the same time. The following check avoids
                                    018B    604          ; a second process from messing up the context of the first
                                    018B    605          ; by avoiding the buffering code.
                                    018B    606          ;
                                    018B    607 170$:
            50    0124 C5     D0  018B    608          MOVL    UCB$L_CT_PID(R5),R0      ; save last writer PID
         0124 C5    60 A4     D0  0190    609          MOVL    PCB$L_PID(R4),UCB$L_CT_PID(R5) ; set new writer PID
            0124 C5    50     D1  0196    610          CMPL    R0,UCB$L_CT_PID(R5)      ; compare PIDs
                        EB    12  019B    611          BNEQ    150$                     ; Branch if not equal
```

```
                03   E2  019D   612          BBSS    #FLG$V_INWRTFDT,-
          E5 00DC C5       019F   613                  UCB$W_CT_FLAGS(R5),150$ ; Set "in FDT" flag, check for busy
                           01A3   614          ;
                           01A3   615          ; R8 will be data + protocol overhead size
                           01A3   616          ;
                           01A3   617 300$:
                07   C1  01A3   618          ADDL3   #<CTP$T_WR_DATA-CTP$W_MSGSIZE>,-
             58    57       01A5   619                  R7,R8                   ; Size of data and protocol overhead
          52    00FC C5   D0  01A7   620          MOVL    UCB$L_CT_WRTCUR(R5),R2  ; fetch current buffer
                23   13  01AC   621          BEQL    310$                    ; Branch if none
                           01AE   622          ;
                           01AE   623          ; There is a buffer that is being filled. See if the current
                           01AE   624          ; data fits into it.
                           01AE   625          ;
          58    0100 C5   B1  01AE   626          CMPW    UCB$W_CT_WRTSIZ(R5),R8  ; Is there room?
                40   1E  01B3   627          BGEQU   320$                    ; Branch if there is room
                           01B5   628          ;
                           01B5   629          ; Not enough room in current buffer, flush current buffer and
                           01B5   630          ; start with a new one.
                           01B5   631          ;
             0118 C5   9E  01B5   632          movab   ucb$l_cnt_ovr(r5),-     ; %%% performance hook
             0120 C5       01B9   633                  ucb$l_cnt_addr(r5)      ; set address to count if flush occurs
                0B21   30  01BC   634          BSBW    CT_WRITE_QRTCTP         ; Get rid of the thing
             011C C5   9E  01BF   635          movab   ucb$l_cnt_frc(r5),-     ; %%% performance hook
             0120 C5       01C3   636                  ucb$l_cnt_addr(r5)      ; reset address to count if flush occurs
               08 50   E8  01C6   637          BLBS    R0,310$
                  08   8A  01C9   638          BICB    #FLG$M_INWRTFDT,-
             00DC C5       01CB   639                  UCB$W_CT_FLAGS(R5)      ; OK if timer fires now
                0757   31  01CE   640          BRW     FDT_ABORT               ; abort, net must have hung up
                           01D1   641          ;
                           01D1   642          ; Allocate new buffer and initialize fields
                           01D1   643          ;
                           01D1   644 310$:
          51    0104 C5   3C  01D1   645          MOVZWL  UCB$W_CT_MAXMSG(R5),R1  ; Set size to allocate
            7E    30 A3   3C  01D6   646          MOVZWL  IRP$W_BOFF(R3),-(SP)    ; Save BOFF
                  075C   30  01DA   647          BSBW    FDT_ALLOC_MESSAGE       ; get the buffer
                     51 8ED0  01DD   648          POPL    R1                      ; Fetch old BOFF
               30 A3    51   A0  01E0   649          ADDW    R1,IRP$W_BOFF(R3)       ; Add to quota charged
          00F8 C5    52   D0  01E4   650          MOVL    R2,UCB$L_CT_WRTCTP(R5)  ; Set address
          00FC C5    52   D0  01E9   651          MOVL    R2,UCB$L_CT_WRTCUR(R5)  ; Set address
0100 C5     08 A2    28   A3  01EE   652          SUBW3   #CTP$W_MSGSIZE, -
                           01F5   653                  CTP$W_SIZE(R2), -
                           01F5   654                  UCB$W_CT_WRTSIZ(R5)     ; Size - (nonpaged and foundation overhead)
                           01F5   655 320$:
             0102 C5   B6  01F5   656          INCW    UCB$W_CT_WRTCNT(R5)     ; Increment use count
                           01F9   657          ;
                           01F9   658          ; Fill in buffer
                           01F9   659          ;
             50    01   3C  01F9   660          MOVZWL  #1,R0                   ; signal buffering
                0078   30  01FC   661          BSBW    CT_WRITE_FILLIN         ; call fill in routine
                           01FF   662          ;
                           01FF   663          ; Now update UCB field that describe buffer
                           01FF   664          ;
          0100 C5    58   A2  01FF   665          SUBW2   R8,UCB$W_CT_WRTSIZ(R5)  ; Room left
          00FC C5    58   C0  0204   666          ADDL2   R8,UCB$L_CT_WRTCUR(R5)  ; Add in offset
                           0209   667          ;
                           0209   668          ; See if entire buffer was copied. If done, exit,
```

```
                     0209   669          ; if not, then queue packet and try again.
                     0209   670          ;
          57   D5    0209   671          TSTL    R7                      ; Done?
          06   13    020B   672          BEQL    350$                    ; Branch if yes
       59   10   8A  020D   673          BICB    #CTP$M_WR_BEGIN,R9      ; Clear "start of message"
        FF90   31    0210   674          BRW     300$                    ; Loop
                     0213   675          ;
                     0213   676          ; Finish up and Exit QIO
                     0213   677          ;
                     0213   678 350$:
          20   88    0213   679          BISB    #CTP$M_WR_END,-
        2B A2        0215   680                  CTP$W_WR_FLAGS(R2)      ; Set "end of message"
        2C A3   D4   0217   681          CLRL    IRP$L_SVAPTE(R3)        ; Clear pointer
         0099   30    021A   682          BSBW    SETUP_TQE               ; Start timer...
   38 A3   01   B0    021D   683          MOVW    #SS$_NORMAL,IRP$L_IOST1(R3)     ; status
3A A3   32 A3   B0    0221   684          MOVW    IRP$W_BCNT(R3),IRP$L_IOST1+2(R3); byte count
        3C A3   D4    0226   685          CLRL    IRP$L_IOST2(R3)         ; no cursor position or lines output
          08   8A    0229   686          BICB    #FLG$M_INWRTFDT,-
       00DC C5        022B   687                  UCB$W_CT_FLAGS(R5)      ; OK if timer fires now
   33  00C0 C5   E8   022E   688          BLBS    UCB$L_CT_RIIRP(R5),500$ ; branch if net hung up
          03   B1    0233   689          CMPW    #CT$K_CTPQLIM,-
       00DE C5        0235   690                  UCB$W_CT_QCTPCNT(R5)    ; Queued CTP's >= limit?
          0B   18    0238   691          BGEQ    360$                    ; Branch if no
                     023A   692          ;
                     023A   693          ; Many CTP's queued off UCB, don't complete this write now.
                     023A   694          ;
 00F4 D5    63   0E  023A   695          INSQUE  (R3),@UCB$L_CT_STALLQBL(R5) ; Insert in stalled queue
  00000000'GF   17   023F   696          JMP     G^EXE$QIORETURN         ; Return from the qio
                     0245   697 360$:
   50   38 A3   D0   0245   698          MOVL    IRP$L_IOST1(R3),R0      ; Set status
         06CC   31    0249   699 370$:   BRW     FDT_FINISHIOC           ; Finish write to user NOW!
                     024C   700
                     024C   701          ;
                     024C   702          ; Non-buffered writes. Allocate CTP that is exactly the right size.
                     024C   703          ;
                     024C   704 400$:
51   57    2F   C1   024C   705          ADDL3   #CTP$C_WR_LEN,R7,R1     ; Add header to request size
         06E6   30    0250   706          BSBW    FDT_ALLOC_MESSAGE       ; Allocate the message buffer
                     0253   707
59   0420 8F   A8   0253   708          BISW    #<CTP$M_WR_STATUS!-
                     0258   709                  CTP$M_WR_END>,R9        ; Set status and end of msg
  40 A3    08   90   0258   710          MOVB    #CTP$C_MT_WRITE_COM,-
                     025C   711                  IRP$B_CT_RESPTYPE(R3)   ; Set response type expected
          50   D4    025C   712          CLRL    R0                      ; signal not buffering
          17   10    025E   713          BSBB    CT_WRITE_FILLIN         ; fill in message
                     0260   714          ;
                     0260   715          ; Send the message to the remote device and exit QIO service
                     0260   716          ;
       52   51   D0   0260   717          MOVL    R1,R2                   ; Pointer beyond data in message
         0AB1   31    0263   718          BRW     CT_NETMSGSENDX          ;
                     0266   719
                     0266   720          ;
                     0266   721          ; NET has hung up during write formatting, abort write and CTP
                     0266   722          ;
                     0266   723 500$:
       00F8 C5   D0   0266   724          MOVL    UCB$L_CT_WRTCTP(R5),-
         2C A3        026A   725                  IRP$L_SVAPTE(R3)        ; Move wrtctp to IRP, let iopost delete
```

```
                00F8 C5   7C   026C   726          CLRQ    UCB$L_CT_WRTCTP(R5)     ; clear pointers
          50    20E4 8F   3C   0270   727          MOVZWL  #SS$_CINRABORT,R0       ; set status
                     D2   11   0275   728          BRB     370$                    ; exit
                               0277   729
```

```
                     0277   731 .SBTTL  CT_WRITE_FILLIN - Fill buffer with write protocol
                     0277   732
                     0277   733 CT_WRITE_FILLIN:
                     0277   734
                     0277   735 ;
                     0277   736 ; inputs:
                     0277   737 ;
                     0277   738 ;       R0 - lbs if single shot output, lbc if buffered
                     0277   739 ;       R2 - CTP
                     0277   740 ;       R5 - UCB
                     0277   741 ;       R6 - address of user buffer
                     0277   742 ;       R7 - length of user buffer
                     0277   743 ;       R8 - size of data + overhead
                     0277   744 ;
                     0277   745 ; outputs:
                     0277   746 ;
                     0277   747 ;       R1 - last byte filled in
                     0277   748 ;       R6 - updated user buffer address
                     0277   749 ;       R7 - number of bytes remaining in user buffer
                     0277   750 ;       R8 - number of bytes actually written in this call
                     0277   751 ;
          3C   BB    0277   752         PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
    53    28 A2   9E 0279   753         MOVAB   CTP$W_MSGSIZE(R2),R3    ; Base of write protocol message
          56   DD    027D   754         PUSHL   R6                      ; Save Address
       51 57   D0    027F   755         MOVL    R7,R1                   ; Assume size for MOVC
                     0282   756
        1A 50   E9   0282   757         BLBC    R0,20$                  ; Branch if not buffering
52    0100 C5   3C   0285   758         MOVZWL  UCB$W_CT_WRTSIZ(R5),R2  ; Fetch size of buffer remaining
       52 58   D1    028A   759         CMPL    R8,R2                   ; Is data larger than buffer?
          06   1B    028D   760         BLEQU   10$                     ; Branch if it fits
                     028F   761         ;
                     028F   762         ; Doesn't all fit, will have to do multiple buffers
                     028F   763         ;
51    52   D0    028F   764         MOVL    R2,R1                   ; Set size to copy
51    07   C2    0292   765         SUBL2   #<CTP$T_WR_DATA-CTP$W_MSGSIZE>,R1 ; minus overhead
57    51   C2    0295   766 10$:     SUBL2   R1,R7                   ; this is how much will be left
56    51   C0    0298   767         ADDL2   R1,R6                   ; update address
      07   C1    029B   768         ADDL3   #<CTP$T_WR_DATA-CTP$W_MSGSIZE>,-
58    51         029D   769                 R1,R8                   ; data + overhead transferred
                     029F   770 20$:
                     029F   771         ;
                     029F   772         ; Fill in buffer using autoincrement, assumes follow
                     029F   773         ;
                     029F   774         ASSUME  CTP$B_MSGTYPE    EQ CTP$W_MSGSIZE+2
                     029F   775         ASSUME  CTP$W_WR_FLAGS   EQ CTP$B_MSGTYPE+1
                     029F   776         ASSUME  CTP$B_WR_PREFIX  EQ CTP$W_WR_FLAGS+2
                     029F   777         ASSUME  CTP$B_WR_POSTFIX EQ CTP$B_WR_PREFIX+1
                     029F   778         ASSUME  CTP$T_WR_DATA    EQ CTP$B_WR_POSTFIX+1
                     029F   779
      05   A1    029F   780         ADDW3   #<CTP$T_WR_DATA-CTP$B_MSGTYPE>,-
83    51         02A1   781                 R1,(R3)+                ; Set message size
83    07   90    02A3   782         MOVB    #CTP$C_MT_WRITE,(R3)+   ; Set message type
83    59   B0    02A6   783         MOVW    R9,(R3)+                ; Copy flags
83    5A   B0    02A9   784         MOVW    R10,(R3)+               ; Copy prefix/postfix
                     02AC   785         ;
                     02AC   786         ; Header is now set, data remains. If not buffering, it is
                     02AC   787         ; assumed here that the buffer was allocated to be large
```

```
                        02AC    788         ; enough for the one QIO. In other words, buffering is the
                        02AC    789         ; only way that large QIO's will be broken up into smaller
                        02AC    790         ; net packets.
                        02AC    791         ;
 63   9E   51   28     02AC    792         MOVC3   R1,@(SP)+,(R3)          ; Copy buffer
      51   53   D0     02B0    793         MOVL    R3,R1                   ; Address of next byte
           3C   BA     02B3    794         POPR    #^M<R2,R3,R4,R5>        ; Restore registers
                05     02B5    795         RSB                             ; return
                       02B6    796
```

```
                        02B6   798 .SBTTL  SETUP_TQE - Set up and queue TQE
                        02B6   799
                        02B6   800 ;
                        02B6   801 ; R5 - UCB
                        02B6   802 ;
                        02B6   803 ; R0-R2 scratch
                        02B6   804 ;
                        02B6   805
                        02B6   806 SETUP_TQE:
                        02B6   807
                        02B6   808          DSBINT  #IPL$_TIMER                 ; Set IPL
     52    00E4 C5     D0 02BC   809          MOVL    UCB$L_CT_TQE(R5),R2         ; Fetch Timer Queue Entry
           2F          13 02C1   810          BEQL    30$                         ; Branch if one does not exist
                       02C3   811                                              ; (because of error allocating
                       02C3   812                                              ;  or hangup in progress)
           00          E0 02C3   813          BBS     #TQE$V_BSY,-
        2A 10 A2       02C5   814                  TQE$L_FR3(R2),30$           ; Branch if busy (already queued)
                       02C8   815          ;
                       02C8   816          ; Queue TQE
                       02C8   817          ;
           38          BB 02C8   818          PUSHR   #^M<R3,R4,R5>               ; Save
         0102 C5       B4 02CA   819          CLRW    UCB$W_CT_WRTCNT(R5)         ; Clear use count
       14 A2    55     D0 02CE   820          MOVL    R5,TQE$L_FR4(R2)            ; Set UCB address
          55    52     D0 02D2   821          MOVL    R2,R5                       ; Set TQE as input
     50 00000000'GF    7D 02D5   822          MOVQ    G^EXE$GQ_SYSTIME,R0         ; Fetch current time
                       02DC   823
     50 001E8480 8F    C0 02DC   824          ADDL    #CT$K_TQE_DELTA,R0          ; Add delta
           51    00    D8 02E3   825          ADWC    #0,R1                       ; Add with carry into second part
       10 A2     01    88 02E6   826          BISB    #TQE$M_BSY,TQE$L_FR3(R2)    ; Set as busy
     00000000'GF       16 02EA   827          JSB     G^EXE$INSTIMQ               ; Insert in queue
           38          BA 02F0   828          POPR    #^M<R3,R4,R5>               ; Restore
                       02F2   829 30$:
                       02F2   830          ENBINT                              ; Reset IPL
           05          05 02F5   831          RSB                                 ; Exit
                       02F6   832
```

```
                                                I 13
                     02F6   834 .SBTTL  TQE_WAKEUP - timer wakeup routine
                     02F6   835 TQE_WAKEUP:
                     02F6   836 ;
                     02F6   837 ; R3 - flags
                     02F6   838 ; R4 - UCB
                     02F6   839 ; R5 - TQE
                     02F6   840 ; IPL = IPL$TIMER (IPL$_SYNCH)
                     02F6   841 ;
                     02F6   842 ; On return:
                     02F6   843 ;
                     02F6   844 ; R0-R4 are scratch, R5 must point to a valid TQE
                     02F6   845 ;
                     02F6   846
        01   90      02F6   847          MOVB    #TQE$C_SSSNGL,-
          0B A5      02F8   848                  TQE$B_RQTYPE(R5)          ; Set request type as system subroutine
10 A5     01   8A    02FA   849          BICB    #TQE$M_BSY,TQE$L_FR3(R5) ; clear busy flag
                     02FE   850
33 53     01   E0    02FE   851          BBS     #TQE$V_DELETE,R3,30$     ; if terminal has hungup, delete TQE
          03   E0    0302   852          BBS     #FLG$V_INWRTFDT,-
2C 00DC   C4         0304   853                  UCB$W_CT_FLAGS(R4),20$   ; if FDT in progress, exit
                     0308   854          ;
                     0308   855          ; If a write has occured since being queued, simply reschedule TQE
                     0308   856          ;
    0102 C4    B5    0308   857          TSTW    UCB$W_CT_WRTCNT(R4)      ; Test count
          0D   13    030C   858          BEQL    10$                      ; zero implies no new qio's
                     030E   859
    0102 C4    B4    030E   860          CLRW    UCB$W_CT_WRTCNT(R4)      ; Clear use count
          05   90    0312   861          MOVB    #TQE$C_SSREPT,-
          0B A5      0314   862                  TQE$B_RQTYPE(R5)          ; Set request type as repeating
10 A5     01   88    0316   863          BISB    #TQE$M_BSY,TQE$L_FR3(R5) ; Set busy flag
               05    031A   864          RSB
                     031B   865          ;
                     031B   866          ; TQE will now "fire" the write to the net
                     031B   867          ;
                     031B   868 10$:
             55   DD 031B   869          PUSHL   R5                       ; Save TQE address
          55 54   D0 031D   870          MOVL    R4,R5                    ; Set UCB address
        0114 C5   9E 0320   871          movab   ucb$l_cnt_tqe(r5),-      ; %%% performance hook
        0120 C5      0324   872                  ucb$l_cnt_addr(r5)       ; set address to count if flush occurs
                     0327   873
           09B6   30 0327   874          BSBW    CT_WRITE_WRTCTP          ; Flush write buffer if one exists
                     032A   875
        011C C5   9E 032A   876          movab   ucb$l_cnt_frc(r5),-      ; %%% performance hook
        0120 C5      032E   877                  ucb$l_cnt_addr(r5)       ; reset address to count if flush occurs
             55 8ED0 0331   878          POPL    R5                       ; Must restore TQE address before
                     0334   879 20$:
                05   0334   880          RSB                              ;  returning
                     0335   881          ;
                     0335   882          ; Terminal has hungup while TQE was queued, delete TQE now
                     0335   883          ;
                     0335   884 30$:
             50   55 D0 0335   885          MOVL    R5,R0                    ; TQE address
     00000000'GF   16 0338   886          JSB     G^EXE$DEANONPAGED        ; Deallocate the I/O packet (DRVDEALMEM?)
55   00000000'GF   DE 033E   887          MOVAL   G^EXE$AL_TQENOREPT,R5    ; Dummy TQE (needed for return to system)
                05   0345   888          RSB                              ; Return
```

```
                     0346   890          .SBTTL  CT_READ - Function Decision Routine for READ Functions
                     0346   891
                     0346   892 ;++
                     0346   893 ; CT_READ - Function Decision Routine for READ Functions
                     0346   894 ;
                     0346   895 ; Functional description:
                     0346   896 ;
                     0346   897 ;       This routine is called by the SYS$QIO service to dispatch a READ
                     0346   898 ;       I/O request.
                     0346   899 ;
                     0346   900 ;       The QIO parameters for terminal READS are:
                     0346   901 ;
                     0346   902 ;       P1 = address of the buffer
                     0346   903 ;       P2 = size of the buffer
                     0346   904 ;       P3 = number of seconds to wait for characters
                     0346   905 ;       P4 = address of terminator class bitmask or 0 if standard
                     0346   906 ;       P5 = address of prompt string for IO$_READPROMPT or IO$_TTYREADPALL
                     0346   907 ;       P6 = size of prompt string for IO$_READPROMPT or IO$_TTYREADPALL
                     0346   908 ;
                     0346   909 ;       The buffer is validated for access, the process's quota checked and
                     0346   910 ;       decremented, the timeout, terminator mask, and prompt are copied to a
                     0346   911 ;       message block, the address of the message block is stored in the IRP,
                     0346   912 ;       and the IRP is queued to the ACP for delivery to the remote system.
                     0346   913 ;
                     0346   914 ; Inputs:
                     0346   915 ;
                     0346   916 ;       R0-R2 = scratch registers
                     0346   917 ;       R3 = address of the IRP (I/O request packet)
                     0346   918 ;       R4 = address of the PCB (process control block)
                     0346   919 ;       R5 = address of the UCB (unit control block)
                     0346   920 ;       R6 = address of the CCB (channel control block)
                     0346   921 ;       R7 = bit number of the I/O function code
                     0346   922 ;       R8 = address of the FDT table entry for this routine
                     0346   923 ;       R9-R11 = scratch registers
                     0346   924 ;       AP = address of the 1st function dependent QIO parameter
                     0346   925 ;
                     0346   926 ; Outputs:
                     0346   927 ;
                     0346   928 ;       IRP$L_SVAPTE(R3) = address of message buffer
                     0346   929 ;       IRP$W_BOFF(R3) = size of message buffer
                     0346   930 ;       IRP$L_MEDIA(R3) = address of user buffer
                     0346   931 ;       IRP$W_BCNT(R3) = size of user buffer
                     0346   932 ;
                     0346   933 ;       The routine preserves all registers except R0-R2, and
                     0346   934 ;       R6-R11.
                     0346   935 ;
                     0346   936 ;--
                     0346   937
                     0346   938 ; Local storage offsets on stack:
                     0346   939
                     0346   940 .SAVE_LOCAL_BLOCK                    ; save psect
                     0346   941 .PSECT $ABS$,ABS                     ; set null psect
                     0138   942
00000000             0138   943 . = 0                                ; Initial offset
                     0000   944
00000004             0000   945 bufaddr:        .BLKL                ; Input buffer
00000008             0004   946 bufsize:        .BLKL                ;
```

```
                                                K 13
0000000C  0008    947 prmaddr:        .BLKL           ; Prompt buffer
00000010  000C    948 prmsize:        .BLKL           ;
00000014  0010    949 trmaddr:        .BLKL           ; Terminator Set buffer
00000018  0014    950 trmsize:        .BLKL           ;
0000001C  0018    951 iniaddr:        .BLKL           ; Initial String buffer
00000020  001C    952 inisize:        .BLKL           ;
00000024  0020    953 altechaddr:     .BLKL           ; Alternate echo string buffer (READ VERIFY)
00000028  0024    954 altechsize:     .BLKL           ;
0000002C  0028    955 picstraddr:     .BLKL           ; Picture string buffer (READ VERIFY)
00000030  002C    956 picstrsize:     .BLKL           ;
00000034  0030    957 timeout:        .BLKL           ; timeout in seconds
00000038  0034    958 inioffset:      .BLKL           ; initial string offset
0000003A  0038    959 editflags:      .BLKW           ; Edit flags (READ VERIFY)
0000003C  003A    960 fillchar:       .BLKW           ; fill characters (READ VERIFY)
0000003E  003C    961 editmode:       .BLKW           ; Edit mode (READ VERIFY)
          003E    962
0000003E  003E    963 read_local = .                  ; overall size
          003E    964
          00000346  965 .RESTORE                      ; restore psect
              0346  966
              0346  967 ;
              0346  968 ; Register Usage:
              0346  969 ;
              0346  970 ; R0,R1 - temporary scratch
              0346  971 ; R2    - CTP
              0346  972 ; R3    - IRP
              0346  973 ; R4    - PCB
              0346  974 ; R5    - UCB
              0346  975 ; R6    -
              0346  976 ; R7    - function code
              0346  977 ; R8    - local storage pointer
              0346  978 ; R9    - flags
              0346  979 ; R10   -
              0346  980 ; R11   -
              0346  981 ;
              0346  982
              0346  983 CT_READ:                          ; READ FDT routine
              0346  984
        42 A3   B4    0346  985         CLRW    IRP$W_CT_POST(R3)       ; Clear POSTed flag
           02   AA    0349  986         BICW    #FLG$M_CTRLO,-
      00DC C5         034B  987                 UCB$W_CT_FLAGS(R5)      ; Set no control O
         098F   30    034E  988         BSBW    CT_WRITE_WRTCTP         ; See if flush must happen
                      0351  989         ;
                      0351  990         ; Allocate and zero local storage
                      0351  991         ;
        5E   3E   C2  0351  992         SUBL2   #READ_LOCAL,SP          ; Allocate local storage
        58   5E   D0  0354  993         MOVL    SP,R8                   ; Save pointer
             38   BB  0357  994         PUSHR   #^M<R3,R4,R5>           ; save registers
00      6E   00   2C  0359  995         MOVC5   #0,(SP),#0,-
        68   3E        035D  996                 #READ_LOCAL,(R8)        ; zero local storage
             38   BA  035F  997         POPR    #^M<R3,R4,R5>           ; restore registers
                      0361  998
59    9008 8F   3C    0361  999         MOVZWL  #CTP$M_SR_FORMAT!-      ; Assume formatting and
                      0366 1000                  CTP$M_SR_NORMTERM!-    ;  normal terminators and
                      0366 1001                  CTP$M_SR_TRMECHO,R9    ;  normal terminator echo
                      0366 1002         ;
                      0366 1003         ; Check access to user's buffer
```

```
                     0366  1004          ;
                     0366  1005          MOVL    P1(AP),R0                       ; Get user buffer virtual address
                     0369  1006          MOVL    R0,IRP$L_MEDIA(R3)              ; Save address in packet
                     036D  1007          MOVZWL  P2(AP),R7                       ; Get buffer size
                     0371  1008          BEQL    5$                              ; Skip check if zero
                     0373  1009          MOVQ    R0,BUFADDR(R8)                  ; Save address and length
                     0376  1010          JSB     G^EXE$READCHK                   ; Check buffer access
                     037C  1011                                                  ; (no return means no access)
                     037C  1012 5$:
                     037C  1013          CMPW    BUFSIZE(R8),-
                     037F  1014                  UCB$W_CT_MAXREAD(R5)            ; Size of read server can do
                     0382  1015          BLEQU   10$                             ; branch if ok
                     0384  1016 ;**       MOVZWL  #SS$_IVBUFLEN,R0                ; ** status
                     0384  1017 ;**       BRW     EXIT_ABORT                      ; exit
                     0384  1018          MOVW    UCB$W_CT_MAXREAD(R5),-          ; Force size of read server can do
                     0388  1019                  BUFSIZE(R8)
                     038A  1020          ;
                     038A  1021          ; Check for extended item list format
                     038A  1022          ;
                     038A  1023 10$:
                     038A  1024          BBC     #IO$V_EXTEND,-
                     038C  1025                  IRP$W_FUNC(R3),15$               ; branch if not item list format
                     038F  1026          BSBW    CT_READ_ITMLST                  ; Format data
                     0392  1027          BRW     200$                            ; continue
                     0395  1028          ;
                     0395  1029          ; Check I/O function code for mode of read
                     0395  1030          ;
                     0395  1031 15$:
                     0395  1032          CMPB    #IO$_READPBLK,R7                ; Function code IO$_READPBLK?
                     0398  1033          BEQL    20$                             ; Branch if yes
                     039A  1034          CMPB    #IO$_TTYREADALL,R7              ; Function code IO$_TTYREADALL?
                     039D  1035          BNEQ    30$                             ; Branch if no
                     039F  1036 20$:     BISW    #CTP$M_SR_ALLBUTX,R9            ; Set read all but XON/XOFF in flags
                     03A4  1037 30$:
                     03A4  1038          ;
                     03A4  1039          ; Now check for prompting
                     03A4  1040          ;
                     03A4  1041          CMPB    #IO$_READPROMPT,R7              ; Read prompt?
                     03A7  1042          BEQL    40$                             ; Branch if yes
                     03A9  1043          CMPB    #IO$_TTYREADPALL,R7             ; Read passall with prompt?
                     03AC  1044          BNEQ    50$                             ; Branch if not
                     03AE  1045          BISW    #CTP$M_SR_ALLBUTX,R9            ; Set read all but XON/XOFF in flags
                     03B3  1046 40$:     MOVZWL  P6(AP),R1                       ; Get size of prompt
                     03B7  1047          BEQL    50$                             ; If eql then make this normal read
                     03B9  1048          MOVL    P5(AP),R0                       ; Get prompt buffer address
                     03BD  1049 ;
                     03BD  1050 ; Check access to prompt string
                     03BD  1051 ;
                     03BD  1052          MOVQ    R0,PRMADDR(R8)                  ; save address and size of prompt
                     03C1  1053          JSB     G^EXE$WRITECHK                  ; Check prompt access
                     03C7  1054                                                  ; (no return means no access)
                     03C7  1055 ;
                     03C7  1056 ; Get terminator bitmask and check access
                     03C7  1057 ;
                     03C7  1058 50$:
                     03C7  1059          CLRL    R2                              ; Assume no terminator specified
                     03C9  1060          MOVL    P4(AP),R1                       ; Get address of terminator desc
```

```
                     2B   13   03CD  1061          BEQL    65$                     ; If eql none specified
                50   0C   3C   03CF  1062          MOVZWL  #SS$_ACCVIO,R0          ; Assume no access
                               03D2  1063          IFNORD  #8,(R1),62$             ; Descriptor accessible?
                52   61   3C   03D8  1064          MOVZWL  (R1),R2                 ; Get bitmask size
                     08   12   03DB  1065          BNEQ    60$                     ; If neq long format
                52   04   D0   03DD  1066          MOVL    #4,R2                   ; Size of short format
                51   04   C0   03E0  1067          ADDL    #4,R1                   ; Set address of bitmask
                     15   11   03E3  1068          BRB     65$                     ;
                               03E5  1069 60$:
            51   04 A1   D0   03E5  1070          MOVL    4(R1),R1                ; Get address of long format bitmask
                               03E9  1071          IFNORD  R2,(R1),62$             ; Bitmask accessible?
                20   52   B1   03EF  1072          CMPW    R2,#32                  ; Bitmask greater than allowed size?
                     06   1B   03F2  1073          BLEQU   65$                     ; If less than or equal, no
                50   14   3C   03F4  1074          MOVZWL  #SS$_BADPARAM,R0        ; Set status
                   052E   31   03F7  1075 62$:     BRW     FDT_ABORT               ; branch to read error
                               03FA  1076 65$:
            10 A8   51   7D   03FA  1077          MOVQ    R1,TRMADDR(R8)          ; save address and size of terminators
                               03FE  1078 ;
                               03FE  1079 ; Set up read flags
                               03FE  1080 ;
                50   20 A3   3C   03FE  1081          MOVZWL  IRP$W_FUNC(R3),R0       ; Fetch function code and modifiers
                     01A8   30   0402  1082          BSBW    FDT_READFLAGS           ; map bits to TSA
            30 A8   08 AC   D0   0405  1083          MOVL    P3(AP),TIMEOUT(R8)      ; Copy timeout parameter
                               040A  1084 ;
                               040A  1085 ; Allocate the message buffer
                               040A  1086 ;
                               040A  1087 200$:
                04 A8   B0   040A  1088          MOVW    BUFSIZE(R8),-
                32 A3        040D  1089                  IRP$W_BCNT(R3)          ; reset BCNT written by EXE$WRITECHK
                               040F  1090          ;
                               040F  1091          ; Calculate size and allocate CTP
                               040F  1092          ;
              57   0C A8   D0   040F  1093          MOVL    PRMSIZE(R8),R7          ; Set prompt size
            51   57   3B   C1   0413  1094          ADDL3   #CTP$C_SR_LEN,R7,R1     ; Add header size to prompt size
              51   1C A8   C0   0417  1095          ADDL2   INISIZE(R8),R1          ; Add in initial string size
              51   14 A8   C0   041B  1096          ADDL2   TRMSIZE(R8),R1          ; Add in terminator size
                30 A8   B5   041F  1097          TSTW    EDITMODE(R8)            ; non-zero if read verify
                     19   13   0422  1098          BEQL    202$                    ; branch if not read verify
                51   08   C0   0424  1099          ADDL2   #CTP$C_SR2_LEN-CTP$C_SR_LEN,R1  ; Add additional header size
              51   24 A8   C0   0427  1100          ADDL2   ALTECHSIZE(R8),R1       ; Add in alternate echo string size
              51   2C A8   C0   042B  1101          ADDL2   PICSTRSIZE(R8),R1       ; Add in Picture string size
                     11   E0   042F  1102          BBS     #CTP$C_MT_VMS_READVFY,-
            08 0108 C5        0431  1103                  UCB$L_C1_CEGA[MSG(R5),202$ ; branch if read verify allowed
            50   00F4 8F   3C   0435  1104          MOVZWL  #SS$_ILLIOFUNC,R0       ; exit with error
                   04EB   31   043A  1105          BRW     FDT_ABORT               ; nope, illegal I/O
                               043D  1106 202$:
                   04F9   30   043D  1107          BSBW    FDT_ALLOC_MESSAGE       ; Allocate the message buffer
                     02   90   0440  1108          MOVB    #CTP$C_MT_START_RD,-
                2A A2        0442  1109                  CTP$B_MSGTYPE(R2)       ; Set message type
              40 A3   03   90   0444  1110          MOVB    #CTP$C_MT_READ_DATA,-
                               0448  1111                  IRP$B_CT_RESPTYPE(R3)   ; Set response type expected
                               0448  1112 ;
                               0448  1113 ; Copy the timeout, terminator bitmask, and prompt string to the message
                               0448  1114 ;
              2B A2   59   D0   0448  1115          MOVL    R9,CTP$L_SR_FLAGS(R2)   ; Set flags NB: FLAGS is really only 3 bytes
                32 A3   B0   044C  1116          MOVW    IRP$W_BCNT(R3),-
                2E A2        044F  1117                  CTP$W_SR_MAX_LEN(R2)    ; Set requested byte count
```

```
                     30 A8   B0  0451  1118          MOVW    TIMEOUT(R8),-
                     32 A2       0454  1119                  CTP$W_SR_TIMEOUT(R2)    ; Copy timeout value
                             0456  1120
          30 A2  57  1C A8   A1  0456  1121          ADDW3   INISIZE(R8),R7,-
                             045C  1122                  CTP$W_SR_END_DATA(R2)   ; Set end of data
                             045C  1123          ASSUME  CTP$W_SR_END_PRMT+2 EQ CTP$W_SR_STR_DISP
                  34 A2  57  3C  045C  1124          MOVZWL  R7,CTP$W_SR_END_PRMT(R2); Copy length, zero start display
                  50  34 A8  D0  0460  1125          MOVL    INIOFFSET(R8),R0        ; Set initial offset
                         05  13  0464  1126          BEQL    210$                    ; skip if zero
             36 A2  57   50  A1  0466  1127          ADDW3   R0,R7,CTP$W_SR_STR_DISP(R2) ; Set start display
                             046B  1128 210$:
                  38 A2  57  B0  046B  1129          MOVW    R7,CTP$W_SR_LO_WATER(R2); Set Low water mark ***
                         3C  BB  046F  1130          PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
                             0471  1131          ;
                             0471  1132          ; Prompt is different based on start read type (Normal or read verify)
                             0471  1133          ;
                  53  3A A2  9E  0471  1134          MOVAB   CTP$T_SR_TERM(R2),R3    ; Assume normal read
                     3C A8  B5  0475  1135          TSTW    EDITMODE(R8)            ; normal read is zero
                        14  13  0478  1136          BEQL    300$                    ; branch if normal read
                             047A  1137          ;
                             047A  1138          ; Set up read verify additional data
                             047A  1139          ;
                        11   90  047A  1140          MOVB    #CTP$C_MT_VMS_READVFY,-
                     2A A2       047C  1141                  CTP$B_MSGTYPE(R2)       ; Set message type
                             047E  1142          ASSUME  CTP$W_SR2_ALTECHSIZE   EQ CTP$T_SR_TERM
                             047E  1143          ;
                             047E  1144          ; SR2_ALTECHSIZE is the same as SR_TERM, R3 has correct address
                             047E  1145          ;
                             047E  1146          ASSUME  CTP$W_SR2_ALTECHSIZE+2 EQ CTP$W_SR2_PICSTRSIZE
                             047E  1147          ASSUME  CTP$W_SR2_PICSTRSIZE+2 EQ CTP$W_SR2_EDITFLAGS
                             047E  1148          ASSUME  CTP$W_SR2_EDITFLAGS+2  EQ CTP$W_SR2_FILLCHAR
                             047E  1149          ASSUME  CTP$W_SR2_FILLCHAR+2   EQ CTP$T_SR2_TERM
          83   24 A8   B0  047E  1150          MOVW    ALTECHSIZE(R8),(R3)+    ; copy alternate echo size
          83   2C A8   B0  0482  1151          MOVW    PICSTRSIZE(R8),(R3)+    ; copy picture string size
          83   38 A8   B0  0486  1152          MOVW    EDITFLAGS(R8),(R3)+     ; copy flags
          83   3A A8   B0  048A  1153          MOVW    FILLCHAR(R8),(R3)+      ; copy fill characters
                             048E  1154          ;
                             048E  1155          ; fall into starndard read code path...
                             048E  1156          ;
          50   10 A8   7D  048E  1157 230$:     MOVQ    TRMADDR(R8),R0          ; Set addr,size
               83   51   90  0492  1158          MOVB    R1,(R3)+                ; Copy byte count of term mask
                    0A   13  0495  1159          BEQL    300$                    ; Branch if not specified
                    01   F0  0497  1160          INSV    #CTP$C_SR_THISTERM,-
                    0E       0499  1161                  #CTP$V_SR_TERM_SET,-
                    02       049A  1162                  #CTP$S_SR_TERM_SET,-
                  28 A2      049B  1163                  CTP$L_SR_FLAGS(R2)      ; Set "Use this terminator mask"
          63   60   51   28  049D  1164          MOVC3   R1,(R0),(R3)            ; Copy terminator bitmask
                             04A1  1165 300$:
            50   08 A8   7D  04A1  1166          MOVQ    PRMADDR(R8),R0          ; Set addr,size
          63   60   51   28  04A5  1167          MOVC3   R1,(R0),(R3)            ; Copy prompt string
            50   18 A8   7D  04A9  1168          MOVQ    INIADDR(R8),R0          ; Set addr,size
          63   60   51   28  04AD  1169          MOVC3   R1,(R0),(R3)            ; Copy initial string
                             04B1  1170
                  3C A8   B5  04B1  1171          TSTW    EDITMODE(R8)            ; normal read is zero
                     10   13  04B4  1172          BEQL    320$                    ; branch if normal read
            50   28 A8   7D  04B6  1173          MOVQ    PICSTRADDR(R8),R0       ; Set addr,size
          63   60   51   28  04BA  1174          MOVC3   R1,(R0),(R3)            ; Copy initial string
```

```
                50    20 A8    7D   04BE  1175          MOVQ    ALTECHADDR(R8),R0       ; Set addr,size
          63    60    51       28   04C2  1176          MOVC3   R1,(R0),(R3)            ; Copy initial string
                                  04C6  1177 320$:
                51    53       D0   04C6  1178          MOVL    R3,R1                   ; Save addr beyond data
                      3C       BA   04C9  1179          POPR    #^M<R2,R3,R4,R5>        ; Restore registers
                                  04CB  1180 ;
                                  04CB  1181 ; Send the message the remote device and exit the QIO service
                                  04CB  1182 ;
                52    51       D0   04CB  1183          MOVL    R1,R2                   ; Set address beyond data
                   0846        31   04CE  1184          BRW     CT_NETMSGSENDX          ;
```

```
                                04D1  1186          .SBTTL  CT_READ_ITMLST - FDT routine for read with item list
                                04D1  1187 ;++
                                04D1  1188 ;
                                04D1  1189 ;
                                04D1  1190 ;
                                04D1  1191 ; *** a clean up pass is needed to here to verify that the paranoia
                                04D1  1192 ;     checks made by TTDRIVER and this driver are the same.
                                04D1  1193 ;
                                04D1  1194 ;--
                                04D1  1195
                                04D1  1196 CT_READ_ITMLST:
                                04D1  1197
                                04D1  1198          ;
                                04D1  1199          ; Set up probe of itemlist with P3 as access mode
                                04D1  1200          ;
          56    53    D0  04D1  1201          MOVL    R3,R6                   ; Save IRP
50  08 AC    0    00    EF  04D4  1202          EXTZV   #0,#2,P3(AP),R0         ; fetch low 2 bits of parameter
     00000000'GF    16  04DA  1203          JSB     G^EXE$MAXACMODE         ; maximize with mode of caller
          53    50    D0  04E0  1204          MOVL    R0,R3                   ; Set input to probe routine
                          04E3  1205
       50    10 AC    D0  04E3  1206          MOVL    P5(AP),R0               ; Address of itemlist
       51    14 AC    D0  04E7  1207          MOVL    P6(AP),R1               ; size of item list
                 05    13  04EB  1208          BEQL    10$                     ; can't be zero?
          5A    50    7D  04ED  1209          MOVQ    R0,R10                  ; save both
                 09    11  04F0  1210          BRB     30$                     ; ok, continue
          50    14    3C  04F2  1211 10$:     MOVZWL  #SS$_BADPARAM,R0        ; status
          53    56    D0  04F5  1212 20$:     MOVL    R6,R3                   ; Restore IRP
             042D    31  04F8  1213          BRW     FDT_ABORT               ; abort
                          04FB  1214
     00000000'GF    16  04FB  1215 30$:     JSB     G^EXE$PROBER            ; Can it be read?
           F1    50    E9  0501  1216          BLBC    R0,20$                  ; branch if not
          50    5B    D0  0504  1217          MOVL    R11,R0                  ; size
                          0507  1218          ;
                          0507  1219          ; Verify that size is multiple of 12
                          0507  1220          ;
          53    56    D0  0507  1221          MOVL    R6,R3                   ; Restore IRP
                51    D4  050A  1222          CLRL    R1                      ; quadword r0/r1
50  5B    50    0C    7B  050C  1223          EDIV    #12,R0,R11,R0           ; divide
                50    D5  0511  1224          TSTL    R0                      ; must be zero remainder
                DD    12  0513  1225          BNEQ    10$                     ; error
                          0515  1226          ;
                          0515  1227          ; Now loop and conquer item list, item by item
                          0515  1228          ;
                          0515  1229 40$:
          51    8A    3C  0515  1230          MOVZWL  (R10)+,R1               ; Length
          52    8A    3C  0518  1231          MOVZWL  (R10)+,R2               ; item code
          50    8A    D0  051B  1232          MOVL    (R10)+,R0               ; address or immediate value
                8A    D5  051E  1233          TSTL    (R10)+                  ; Must be zero field
                D0    12  0520  1234          BNEQ    10$                     ; error if not
                          0522  1235
                          0522  1236          CASE    R2,-                    ; case on message type
                          0522  1237                  <100$,-                 ; TRM$_MODIFIERS
                          0522  1238                  200$,-                  ; TRM$_EDITMODE
                          0522  1239                  300$,-                  ; TRM$_TIMEOUT
                          0522  1240                  400$,-                  ; TRM$_TERM
                          0522  1241                  500$,-                  ; TRM$_PROMPT
                          0522  1242                  600$,-                  ; TRM$_INISTRING
```

```
                     0522  1243                    700$,-                          ; TRM$_PICSTRING
                     0522  1244                    800$,-                          ; TRM$_FILLCHR
                     0522  1245                    900$,-                          ; TRM$_INIOFFSET
                     0522  1246                    1000$-                          ; TRM$_ALTECHSTR
                     0522  1247                    >,-                             ; TRM$_LASTITM
                     0522  1248                    TYPE = W
                     053A  1249 ;***        ASSUME  TRM$_LASTITM EQ 10              ; Break assembly if not right
          B6   11    053A  1250             BRB     10$
                     053C  1251             ;
                     053C  1252             ; TRM$_MODIFIERS
                     053C  1253             ;
                     053C  1254 100$:
        006E   30    053C  1255             BSBW    FDT_READFLAGS                   ; Handle read flags
          65   11    053F  1256             BRB     200$                            ; Loop
                     0541  1257             ;
                     0541  1258             ; TRM$_EDITMODE
                     0541  1259             ;
                     0541  1260 200$:
    3C A8   50   B0  0541  1261             MOVW    R0,EDITMODE(R8)                 ; Save for use later
          5F   11    0545  1262             BRB     200$                            ; Loop
                     0547  1263             ;
                     0547  1264             ; TRM$_TIMEOUT
                     0547  1265             ;
                     0547  1266 300$:
    30 A8   50   D0  0547  1267             MOVL    R0,TIMEOUT(R8)                  ; Set timeout
59  2000 8F   A8    054B  1268             BISW    #CTP$M_SR_TIMED,R9              ; set read timed bit
          54   11    0550  1269             BRB     200$                            ; loop
                     0552  1270             ;
                     0552  1271             ; TRM$_TERM
                     0552  1272             ;
                     0552  1273 400$:
          51   D5    0552  1274             TSTL    R1                              ; test length
          09   12    0554  1275             BNEQ    410$                            ; If neq long format
       51   04   D0  0556  1276             MOVL    #4,R1                           ; Size of short format
50     F8 AA   9E    0559  1277             MOVAB   -8(R10),R0                      ; Address of immediate data
          13   11    055D  1278             BRB     430$                            ; skip
                     055F  1279 410$:
                     055F  1280             IFNORD  R1,(R0),420$                    ; Bitmask accessible?
       20   51   B1  0565  1281             CMPW    R1,#32                          ; Bitmask greater than allowed size?
          08   1B    0568  1282             BLEQU   430$                            ; If less than or equal, no
          86   11    056A  1283             BRB     10$                             ; bad param *** other status?
       50   0C   3C  056C  1284 420$:       MOVZWL  #SS$_ACCVIO,R0                  ; access violation
          03B6 31    056F  1285             BRW     FDT_ABORT                       ; branch to read error
                     0572  1286 430$:
10 A8   50   7D    0572  1287             MOVQ    R0,TRMADDR(R8)                  ; save address and size of terminators
          2E   11    0576  1288             BRB     200$                            ; continue
                     0578  1289             ;
                     0578  1290             ; TRM$_PROMPT
                     0578  1291             ;
                     0578  1292 500$:
08 A8   50   7D    0578  1293             MOVQ    R0,PRMADDR(R8)                  ; save address and length
          1E   11    057C  1294             BRB     1500$                           ; continue
                     057E  1295             ;
                     057E  1296             ; TRM$_INISTRING
                     057E  1297             ;
                     057E  1298 600$:
18 A8   50   7D    057E  1299             MOVQ    R0,INIADDR(R8)                  ; save address and length
```

```
                 18   11  0582  1300          BRB     1500$                   ; continue
                          0584  1301          ;
                          0584  1302          ; TRM$_PICSTRING
                          0584  1303          ;
                          0584  1304  700$:
          28 A8  50   7D  0584  1305          MOVQ    R0,PICSTRADDR(R8)       ; save address and length
                 12   11  0588  1306          BRB     1500$                   ; continue
                          058A  1307          ;
                          058A  1308          ; TRM$_FILLCHR
                          058A  1309          ;
                          058A  1310  800$:
          3A A8  50   B0  058A  1311          MOVW    R0,FILLCHAR(R8)         ; save for later
                 16   11  058E  1312          BRB     2000$                   ; continue
                          0590  1313          ;
                          0590  1314          ; TRM$_INIOFFSET
                          0590  1315          ;
          34 A8  50   D0  0590  1316  900$:   MOVL    R0,INIOFFSET(R8)        ; Set inioffset
                 10   11  0594  1317          BRB     2000$                   ; continue
                          0596  1318          ;
                          0596  1319          ; TRM$_ALTECHSTR
                          0596  1320          ;
                          0596  1321  1000$:
          20 A8  50   7D  0596  1322          MOVQ    R0,ALTECHADDR(R8)       ; save address and length
                 00   11  059A  1323          BRB     1500$                   ; continue
                          059C  1324          ;
                          059C  1325          ; Check string for access
                          059C  1326          ;
                 51   D5  059C  1327  1500$:  TSTL    R1                      ; no need to probe if zero
                 C6   13  059E  1328          BEQL    2000$                   ; Skip probe
    00000000'GF   16  05A0  1329          JSB     G^EXE$WRITECHK          ; check for read access
                          05A6  1330          ;
                          05A6  1331          ; Loop for next parameter
                          05A6  1332          ;
                          05A6  1333  2000$:
              01 5B   F5  05A6  1334          SOBGTR  R11,2010$               ; Loop
                      05  05A9  1335          RSB                             ; return
               FF68   31  05AA  1336  2010$:  BRW     40$                     ;
```

```
                                05AD  1338 FDT_READFLAGS:
                                05AD  1339
                                05AD  1340 ; Escape is left out of the following optimization because of the frequency
                                05AD  1341 ; of its use.
                                05AD  1342
        50    1BC0 8F    B3     05AD  1343          BITW    #<IO$M_NOECHO!-
                                05B2  1344                    IO$M_TIMED!-
                                05B2  1345                    IO$M_PURGE!-
                                05B2  1346                    IO$M_TRMNOECHO!-
                                05B2  1347                    IO$M_CVTLOW!-
                                05B2  1348                    IO$M_NOFILTR>,R0       ; Optimization - any bits set?
                      34    13  05B2  1349          BEQL    140$                   ; Branch if not
                                05B4  1350
          05 50       06    E1  05B4  1351          BBC     #IO$V_NOECHO,R0,90$    ; Branch if not noecho
        59    0800 8F    A8     05B8  1352          BISW    #CTP$M_SR_NOECHO,R9    ; Set noecho
                                05BD  1353 90$:
          05 50       07    E1  05BD  1354          BBC     #IO$V_TIMED,R0,100$    ; Branch if not timed
        59    2000 8F    A8     05C1  1355          BISW    #CTP$M_SR_TIMED,R9     ; Set timed
                                05C6  1356 100$:
          03 50       0B    E1  05C6  1357          BBC     #IO$V_PURGE,R0,110$    ; Branch if not purge typeahead
              59        04    A8  05CA  1358          BISW    #CTP$M_SR_PURGE,R9     ; Set purge
                                05CD  1359 110$:
          05 50       0C    E1  05CD  1360          BBC     #IO$V_TRMNOECHO,R0,120$ ; Branch if not terminator noecho
        59    1000 8F    AA     05D1  1361          BICW    #CTP$M_SR_TRMECHO,R9   ; Clear terminator echo
                                05D6  1362 120$:
          05 50       08    E1  05D6  1363          BBC     #IO$V_CVTLOW,R0,130$   ; Branch if not CVTLOW
        59    0080 8F    A8     05DA  1364          BISW    #CTP$M_SR_LOWTOUP,R9   ; Set lower to upper case conversion
                                05DF  1365 130$:
          05 50       09    E1  05DF  1366          BBC     #IO$V_NOFILTR,R0,140$  ; Branch if not filter control characters
        59    0200 8F    A8     05E3  1367          BISW    #CTP$M_SR_EDIT,R9      ; Set read edit characters
                                05E8  1368 140$:
          07 50       0E    E1  05E8  1369          BBC     #IO$V_ESCAPE,R0,150$   ; Branch if not escape reaad
      59    00020000 8F    C8     05EC  1370          BISL    #CTP$M_SR_ESCAPE,R9    ; Set read with escape
                                05F3  1371 150$:
                                05F3  1372          ;
                                05F3  1373          ; VMS extension - if VAX to VAX, send extra bits in spare bits.
                                05F3  1374          ;
                      05    E1  05F3  1375          BBC     #FLG$V_VAXTOVAX,-
          0A 00DC C5            05F5  1376                  UCB$W_CT_FLAGS(R5),300$ ; If not VAX to VAX, skip next check
                                05F9  1377          ASSUME  TRM$V_TM_NOEDIT+1 EQ TRM$V_TM_NORECALL
    51  50    02    0F    EF  05F9  1378          EXTZV   #TRM$V_TM_NOEDIT,#2,R0,R1 ; two bits
    59  02    12    51    F0  05FE  1379          INSV    R1,#CTP$V_SR_NOEDIT,#2,R9 ; Stick in here
                                0603  1380 300$:
                                0603  1381          ASSUME  TRM$V_TM_R_JUST+1 EQ TRM$V_TM_AUTO_TAB
    51  50    02    11    EF  0603  1382          EXTZV   #TRM$V_TM_R_JUST,#2,R0,R1 ; two bits
          38 A8       51    B0  0608  1383          MOVW    R1,EDITFLAGS(R8)       ; save
                            05  060C  1384          RSB                            ; return
                                060D  1385
```

```
060D  1387          .SBTTL  CT_SETMODE, Function Decision Routine for SETMODE/SETCHAR
060D  1388 ;++
060D  1389 ; CT_SETMODE, Function Decision Routine for SETMODE/SETCHAR Functions
060D  1390 ;
060D  1391 ; Functional description:
060D  1392 ;
060D  1393 ;       This routine is called by the SYS$QIO service to dispatch a SETMODE
060D  1394 ;       or SETCHAR I/O request.
060D  1395 ;
060D  1396 ;       The QIO parameters for terminal SETMODE or SETCHAR are:
060D  1397 ;
060D  1398 ;
060D  1399 ;               P1 = address of 8 byte characteristics buffer
060D  1400 ;               P2 = 0, 8 or 12
060D  1401 ;               P3 = speed specifier
060D  1402 ;               P4 = fill specifier
060D  1403 ;               P5 = parity flags
060D  1404 ;
060D  1405 ;       IO$V_CTRLYAST -
060D  1406 ;               P1 = AST routine address or zero to cancel
060D  1407 ;
060D  1408 ;       IO$V_CTRLCAST -
060D  1409 ;                P1 = AST routine address or zero to cancel
060D  1410 ;
060D  1411 ;       IO$V_HANGUP -
060D  1412 ;               NONE
060D  1413 ;
060D  1414 ;       The buffer (if any) is validated for access, the process's quota
060D  1415 ;       checked and decremented, a message block is allocated, the parameters
060D  1416 ;       (if any) are stored in the  message block, the address of the message
060D  1417 ;       block is stored in the IRP, and the IRP is queued to the ACP for
060D  1418 ;       delivery to the remote system.
060D  1419 ;
060D  1420 ;       If an AST is to be enabled, an AST control block is allocated locally
060D  1421 ;       hung off the UCB for later delivery upon receipt of a corresponding
060D  1422 ;       attention message from the remote system.
060D  1423 ;
060D  1424 ; Inputs:
060D  1425 ;
060D  1426 ;       R0-R2 = scratch registers
060D  1427 ;       R3 = address of the IRP (I/O request packet)
060D  1428 ;       R4 = address of the PCB (process control block)
060D  1429 ;       R5 = address of the UCB (unit control block)
060D  1430 ;       R6 = address of the CCB (channel control block)
060D  1431 ;       R7 = bit number of the I/O function code
060D  1432 ;       R8 = address of the FDT table entry for this routine
060D  1433 ;       R9-R11 = scratch registers
060D  1434 ;       AP = address of the 1st function dependent QIO parameter
060D  1435 ;
060D  1436 ; Outputs:
060D  1437 ;
060D  1438 ;       IRP$L_SVAPTE(R3) = address of message buffer
060D  1439 ;       IRP$W_BOFF(R3) = size of message buffer
060D  1440 ;
060D  1441 ;       The routine preserves all registers except R0-R2, R7, and R9-R11
060D  1442 ;--
060D  1443 CT_SETMODE:                                ; SETMODE/SETCHAR FDT routine
```

```
                        060D  1444
                  42 A3 B4  060D  1445          CLRW    IRP$W_CT_POST(R3)       ; Clear POSTed flag
                    06CD 30  0610  1446          BSBW    CT_WRITE_WRTCTP         ; See if flush must happen
                         0613  1447
                09    06 EA  0613  1448          FFS     #IO$V_MAINT,#9,-
              51    20 A3     0616  1449                  IRP$W_FUNC(R3),R1       ; Find first set modifier
                         0619  1450          CASE    R1,TYPE=B,LIMIT=#IO$V_MAINT,<-
                         0619  1451                  SET_MAINT,-             ; IO$M_MAINT
                         0619  1452                  SET_CTRLY,-             ; IO$M_CTRLYAST
                         0619  1453                  SET_CTRLC,-             ; IO$M_CTRLCAST
                         0619  1454                  SET_HANGUP,-            ; IO$M_HANGUP
                         0619  1455                  SET_OUTBAND,-           ; IO$M_OUTBAND
                         0619  1456                  SET_CONNECT,-           ; IO$M_CONNECT
                         0619  1457                  SET_DISCONNECT,-        ; IO$M_DISCONNECT
                         0619  1458                  SET_PID,-               ; IO$M_SETPID
                         0619  1459                  SET_BRDCST>             ; IO$M_BRDCST
                         062F  1460 ;
                         062F  1461 ; Set terminal characteristics if CASE falls though
                         062F  1462 ;
           20 A3   FFC0 8F  B3  062F  1463          BITW    #^CIRP$M_FCODE, -       ; Extra bits?
                         0635  1464                  IRP$W_FUNC(R3)          ; as modifier
                      08 13  0635  1465          BEQL    10$                     ; Nope
              50    00F4 8F  3C  0637  1466          MOVZWL  #SS$_ILLIOFUNC, R0      ; Return as illegal operation
                    02E9 31  063C  1467          BRW     FDT_ABORT               ; with an error not success
                         063F  1468 10$:
                    02A8 30  063F  1469          BSBW    GET_PARAMS              ; Fetch P1 and P2 into R1,R2
                         0642  1470          ;
                         0642  1471          ; Send CTERM or VMS message
                         0642  1472          ;
                       0F E1  0642  1473          BBC     #CTP$C_MT_VMSQIO,-
                06 0108 C5     0644  1474                  UCB$L_CT_LEGALMSG(R5),12$ ; Branch if not VMSQIO support
                    F9B5' 30  0648  1475          BSBW    CT_VMS_SETMODE          ; Format VMS message
                    06C9 31   064B  1476          BRW     CT_NETMSGSENDX          ; Send message and exit
                         064E  1477 12$:
                  48 A5 D0    064E  1478          MOVL    UCB$L_DEVDEPND2(R5),-   ;
                       5B     0651  1479                  R11                     ; Extended word is defaulted
                59    81 7D   0652  1480          MOVQ    (R1)+,R9                ; Get characteristics
                0C    52 D1   0655  1481          CMPL    R2, #12                 ; Do we get another longword?
                      03 19   0658  1482          BLSS    20$                     ; Nope
                5B    81 D0   065A  1483          MOVL    (R1)+, R11              ; Obtain the third longword
                         065D  1484 20$:
                         065D  1485 ;
                         065D  1486 ; Map VMS characteristics into TSA
                         065D  1487 ;
           51 000000C8 8F    D0  065D  1488          MOVL    #200,R1                 ; Set size of message buffer *** FIX
                    02D2 30  0664  1489          BSBW    FDT_ALLOC_MESSAGE       ; Allocate a message buffer
                         0667  1490          ASSUME  CTP$B_CH_FLAGS EQ CTP$B_MSGTYPE+1
                       0B 9B  0667  1491          MOVZBW  #CTP$C_MT_CHAR,-
                  2A A2       0669  1492                  CTP$B_MSGTYPE(R2)       ; Set message type
                         066B  1493
                       30 BB  066B  1494          PUSHR   #^M<R4,R5>              ; save
                    F990' 30  066D  1495          BSBW    CT_SET                  ; call external routine
                       30 BA  0670  1496          POPR    #^M<R4,R5>              ; restore
                40 A5  59 7D   0672  1497          MOVQ    R9,UCB$B_DEVCLASS(R5)   ; Set local copy of characteristics
                48 A5  5B D0   0676  1498          MOVL    R11,UCB$L_DEVDEPND2(R5) ; And extended longword
                         067A  1499
             0112 C2   08 AC  B0  067A  1500          MOVW    P3(AP),UCB$W_CT_SPEED(R2) ; Set speed
```

```
                     0680  1501          ASSUME  UCB$B_CT_CRFILL+1 EQ UCB$B_CT_LFFILL
0110 C2   0C AC   B0 0680  1502          MOVW    P4(AP),UCB$B_CT_CRFILL(R2) ; Set fills
0134 C2   10 AC   B0 0686  1503          MOVW    P5(AP),UCB$W_CT_PARITY(R2) ; Set parity
                     068C  1504          ;
                     068C  1505          ; R0 has LBC if there is no new data to tell server about
                     068C  1506          ;
            03 50 E8 068C  1507          BLBS    R0,30$                     ; continue if lbs
              0106 31 068F  1508          BRW     SET_NOP                    ; done
                     0692  1509 30$:
              012C 31 0692  1510          BRW     SET_READY                  ; Send message to net and exit
                     0695  1511
                     0695  1512 SET_CTRLY:
    57  0090 C5   DE 0695  1513          MOVAL   UCB$L_TL_CTRLY(R5),R7      ; Get address of CTRL/Y AST list
5A  02000000 8F   D0 069A  1514          MOVL    #<1@TTY$C_CTRLY>,R10       ; Set ^Y (for later)
           05 E0 06A1  1515          BBS     #FLG$V_VAXTOVAX,-
      03 00DC C5      06A3  1516                  UCB$W_CT_FLAGS(R5),10$     ; If VAX to VAX, ^C enable is implicit
                     06A7  1517          ;
                     06A7  1518          ; For RSX and TOPS, this driver must always enable both ^C and ^Y
                     06A7  1519          ; when ^Y is enabled.
                     06A7  1520          ;
         5A  08   88 06A7  1521          BISB    #<1@TTY$C_CTRLC>,R10       ; Set ^C (for later)
                     06AA  1522 10$:
   00000000'GF   16 06AA  1523          JSB     G^COM$SETATTNAST           ; Enable an attention AST
                     06B0  1524          ;
                     06B0  1525          ; If hangup is set, deliver ^Y ast to immediately with hangup.
                     06B0  1526          ;
 2D 68 A5   03   E1 06B0  1527          BBC     #UCB$V_TT_HANGUP, -
                     06B5  1528                  UCB$W_DEVSTS(R5),CTRL_CY ; Branch if not hangup
      54   57   D0 06B5  1529          MOVL    R7,R4                      ; Set address of AST listhead
      50   64   D0 06B8  1530          MOVL    (R4),R0                    ; get ast block address
1C A0   02CC 8F   3C 06BB  1531          MOVZWL  #SS$_HANGUP,ACB$L_KAST+4(R0) ; Set AST parameter to hangup
           56   DD 06C1  1532          PUSHL   R6                         ; save... needed in FDT?
   56   00A4 C5   D0 06C3  1533          MOVL    UCB$L_TL_CTLPID(R5),R6     ; Set controlling PID
    00000000'GF   16 06C8  1534          JSB     G^COM$DECATTNASTP          ; Deliver the AST immediately
           56 8ED0 06CE  1535          POPL    R6                         ; restore
            00C4 31 06D1  1536          BRW     SET_NOP                    ; Nop this I/O function
                     06D4  1537
                     06D4  1538 SET_CTRLC:
   57   0094 C5   DE 06D4  1539          MOVAL   UCB$L_TL_CTRLC(R5),R7      ; Assume CTRL/C AST enable
      5A   08   D0 06D9  1540          MOVL    #<1@TTY$C_CTRLC>,R10       ; Set ^C (for later)
    00000000'GF   16 06DC  1541          JSB     G^COM$SETATTNAST           ; Enable an attention AST
                     06E2  1542
                     06E2  1543 CTRL_CY:
      59   3B   3C 06E2  1544          MOVZWL  #<CTP$M_CH_OO!-            ; out of band handling
                     06E5  1545                  CTP$M_CH_D!-               ; discard output
                     06E5  1546                  CTP$M_CH_EE>,R9            ; mask
   59   2800 8F   A8 06E5  1547          BISW    #<CTP$M_CH_D!-             ; discard
                     06EA  1548                  <CTP$C_CH_ECHOSTANDARD@CTP$V_CH_EE>- ; echo ^C
                     06EA  1549                  @8>,R9                     ; Set mask
                     06EA  1550          ASSUME  CTP$C_CH_CANCEL EQ 0       ; assume cancel out of band  is zero
           6C   D5 06EA  1551          TSTL    P1(AP)                     ; Enable if non-zero
           05   13 06EC  1552          BEQL    40$                        ; If zero, then message ok
   59   0100 8F   A8 06EE  1553          BISW    #<CTP$C_CH_ICLEAR@8>,R9    ; immediate clear
                     06F3  1554
            00BF 31 06F3  1555 40$:     BRW     FDT_SET_OUTBAND            ; Go format data
                     06F6  1556
                     06F6  1557 ;
```

```
                        06F5  1558 ; Process a setmode for an outofband ast
                        06F6  1559 ;
                        06F6  1560
                        06F6  1561 SET_OUTBAND:
                        06F6  1562
52   0098 C5    9E      06F6  1563          MOVAB   UCB$L_TL_OUTBAND(R5),R2 ; Address of the include mask
57   009C C5    9E      06FB  1564          MOVAB   UCB$L_TL_BANDQUE(R5),R7 ; Address of the out of band list
 00000000'GF    16      0700  1565          JSB     G^COM$SETCTRLAST        ; Enable the asts
                        0706  1566
                        0706  1567          ;
                        0706  1568          ; Now, must step through entire OOB AST list and build
                        0706  1569          ; individual summary masks for INCLUDE, EXCLUDE and ABORT types.
                        0706  1570          ;
                        0706  1571
                00000000  0706  1572          ast_include = 0
                00000004  0706  1573          ast_exclude = 4
                00000008  0706  1574          ast_abort   = 8
                        0706  1575          assume ucb$l_ct_exclude EQ ucb$l_ct_include+4
                        0706  1576          assume ucb$l_ct_abort EQ ucb$l_ct_exclude+4
                        0706  1577
     5E    0C    C2      0706  1578          SUBL    #12,SP                  ; allocate 3 longwords
     5B    5E    D0      0709  1579          MOVL    SP,R11                  ; pointer
        010C    30      070C  1580          BSBW    FETCH_OOB_DATA          ; fetch data from OOB LIST
                        070F  1581
     59    3F    3C      070F  1582          MOVZWL  #<CTP$M_CH_OO!-
                        0712  1583                  CTP$M_CH_I!-
                        0712  1584                  CTP$M_CH_D!-
                        0712  1585                  CTP$M_CH_EE>,R9          ; init mask
                        0712  1586          ASSUME  CTP$C_CH_CANCEL EQ 0    ; assume cancel = 0
                        0712  1587          ASSUME  CTP$C_CH_ECHONONE EQ 0  ; and echo nothing = 0
                        0712  1588          ;
                        0712  1589          ; First, send all the OOB's that have been implicitly disabled
                        0712  1590          ;
57   0128 C5    9E      0712  1591          MOVAB   UCB$L_CT_INCLUDE(R5),R7 ; Base of last enabled OOB's
        6B    CB      0717  1592          BICL3   AST_INCLUDE(R11),-
     5A    67          0719  1593                  AST_INCLUDE(R7),R10      ; new AND NOT old into R10
        04 AB    CB      071B  1594          BICL3   AST_EXCLUDE(R11),-
  50    04 A7          071E  1595                  AST_EXCLUDE(R7),R0       ; new AND NOT old into R10
     5A    50    C8      0721  1596          BISL    R0,R10                  ; add to others
        08 AB    CB      0724  1597          BICL3   AST_ABORT(R11),-
  50    08 A7          0727  1598                  AST_ABORT(R7),R0         ; new AND NOT old into R10
     5A    50    C8      072A  1599          BISL    R0,R10                  ; add to others
           40    10      072D  1600          BSBB    SEND_OOB_MSG            ; send OOB message
                        072F  1601          ;
                        072F  1602          ; Now, check excludes
                        072F  1603          ;
59   0300 8F    A8      072F  1604          BISW    #<CTP$C_CH_HELLO@8>,R9  ; Set hello
  5A    04 AB    D0      0734  1605          MOVL    AST_EXCLUDE(R11),R10    ; Fetch exclude
  04 A7    5A    D1      0738  1606          CMPL    R10,AST_EXCLUDE(R7)     ; compare excludes
           02    13      073C  1607          BEQL    150$                    ; branch if no change
           2F    10      073E  1608          BSBB    SEND_OOB_MSG            ; send message
                        0740  1609          ;
                        0740  1610          ; Now, check includes
                        0740  1611          ;
                        0740  1612 150$:
59   0400 8F    A8      0740  1613          BISW    #<CTP$M_CH_I@8>,R9      ; Set Include bit
     5A    6B    D0      0745  1614          MOVL    AST_INCLUDE(R11),R10    ; Fetch include
```

```
                67   5A   D1  0748  1615          CMPL    R10,AST_INCLUDE(R7)     ; compare includes
                     02   13  074B  1616          BEQL    160$                    ; branch if no change
                     20   10  074D  1617          BSBB    SEND_OOB_MSG            ; send message
                                 074F  1618          ;
                                 074F  1619          ; Now, Check aborts
                                 074F  1620          ;
                                 074F  1621  160$:
            59   0700 8F   AA  074F  1622          BICW    #<CTP$M_CH_I!3@8>,R9    ; Clear Include bit and handling
            59   0900 8F   A8  0754  1623          BISW    #<CTP$C_CH_ICLEAR!-
                                 0759  1624                  CTP$M_CH_D@8>,R9        ; immediate clear and discard output
              5A   08 AB   D0  0759  1625          MOVL    AST_ABORT(R11),R10      ; Fetch aborts
              08 A7   5A   D1  075D  1626          CMPL    R10,AST_ABORT(R7)       ; compare aborts
                     02   13  0761  1627          BEQL    170$                    ; branch if no change
                     0A   10  0763  1628          BSBB    SEND_OOB_MSG            ; send message
                                 0765  1629  170$:
                                 0765  1630          ;
                                 0765  1631          ; Update UCB with new masks
                                 0765  1632          ;
                     6B   7D  0765  1633          MOVQ    AST_INCLUDE(R11),-
                     67           0767  1634                  AST_INCLUDE(R7)         ; include and exclude
                  08 AB   D0  0768  1635          MOVL    AST_ABORT(R11),-
                  08 A7           076B  1636                  AST_ABORT(R7)           ; and abort
                     29   11  076D  1637          BRB     SET_NOP                 ; exit
                                 076F  1638
                                 076F  1639  SEND_OOB_MSG:
                                 076F  1640  ;
                                 076F  1641  ; Send a no-quota-charged out of band message
                                 076F  1642  ;
                                 076F  1643  ; Inputs:
                                 076F  1644  ;
                                 076F  1645  ; R9  = 2 bytes of mask and attributes
                                 076F  1646  ; R10 = Mask of characters to enable/disable
                                 076F  1647  ;
                        5A   D5  076F  1648          TSTL    R10                     ; test mask
                        24   13  0771  1649          BEQL    110$                    ; ignore if zero
                  0818 8F   BB  0773  1650          PUSHR   #^M<R3,R4,R11>          ; Save
                        53   D4  0777  1651          CLRL    R3                      ; set no irp !!!
                     0052   30  0779  1652          BSBW    SETUP_OUTBAND           ; set up buffer
                   14 50    E9  077C  1653          BLBC    R0,100$                 ;
                                 077F  1654          ;
                                 077F  1655          ; CTP returned in R3 !
                                 077F  1656          ;
            51   52   0C A3   C3  077F  1657          SUBL3   CTP$L_MSGDAT(R3),R2,R1  ; Make the length of the data
                 14 A3    51   B0  0784  1658          MOVW    R1,CTP$W_DATSIZE(R3)    ; save in the buffer
                        04   A3  0788  1659          SUBW3   #<CTP$B_MSGTYPE-CTP$B_PRO_MSGTYPE>,-
                 28 A3    51       078A  1660                  R1,CTP$W_MSGSIZE(R3)    ; Fill in size of single message
                     52   53   D0  078D  1661          MOVL    R3,R2                   ; Set address
                        05C7   30  0790  1662          BSBW    CT_NET_Q_MSG            ; Send the message to the server
                                 0793  1663  100$:
                  0818 8F   BA  0793  1664          POPR    #^M<R3,R4,R11>          ; Restore
                                 0797  1665  110$:
                        05       0797  1666          RSB
                                 0798  1667
                                 0798  1668  SET_NOP:
                        017A   31  0798  1669          BRW     FDT_FINISHIOC_OK        ; Complete I/O
                                 079B  1670          ;
                                 079B  1671          ; The following types of modifiers are not allowed on remote terminals
```

```
                     079B  1672          ;
                     079B  1673 SET_MAINT:
                     079B  1674 SET_CONNECT:
                     079B  1675 SET_DISCONNECT:
                     079B  1676
 50  0334 8F   3C    079B  1677          MOVZWL  #SS$_DEVREQERR, R0      ; Return as device request error
       0185    31    07A0  1678          BRW     FDT_ABORT               ; with an error not success
                     07A3  1679
                     07A3  1680 SET_BRDCST:
       0144    30    07A3  1681          BSBW    GET_PARAMS              ; Get parameters
00A8 C5  61    7D    07A6  1682          MOVQ    (R1),UCB$Q_TL_BRKTHRU(R5); Set bits
         EB    11    07AB  1683          BRB     SET_NOP                 ; Set done
                     07AD  1684
                     07AD  1685 SET_PID:
      60 A4    D0    07AD  1686          MOVL    PCB$L_PID(R4),-
      00A4 C5        07B0  1687                  UCB$L_TL_CTLPID(R5)     ; Set controlling PID
         E3    11    07B3  1688          BRB     SET_NOP                 ; done
                     07B5  1689
                     07B5  1690 FDT_SET_OUTBAND:
                     07B5  1691
         17    10    07B5  1692          BSBB    SETUP_OUTBAND           ; set up out of band messag
         08    11    07B7  1693          BRB     SET_READY               ; Send message
                     07B9  1694
                     07B9  1695 SET_HANGUP:
                     07B9  1696
 50      04    9A    07B9  1697          MOVZBL  #UNBIND$C_DISCONNECT,R0 ; unbind reason code
       092D    30    07BC  1698          BSBW    SEND_UNBIND             ; send it with common routine
         D7    11    07BF  1699          BRB     SET_NOP                 ; nop I/O
                     07C1  1700
                     07C1  1701 SET_READY:
         50    7C    07C1  1702          CLRQ    R0                      ; no status other than low word
                     07C3  1703          ASSUME  IRP$L_IOST1+4 EQ IRP$L_IOST2
38 A3    50    7D    07C3  1704          MOVQ    R0,IRP$L_IOST1(R3)      ; Set status for I/O completion
38 A3    01    B0    07C7  1705          MOVW    #SS$_NORMAL, -
                     07CB  1706                  IRP$L_IOST1(R3)         ; Set status
                     07CB  1707
       0549    31    07CB  1708          BRW     CT_NETMSGSENDX          ; Send message and exit
                     07CE  1709
```

```
                                07CE  1711          .SBTTL  SETUP_OUTBAND - Format out of band message
                                07CE  1712
                                07CE  1713 ;+++
                                07CE  1714 ;
                                07CE  1715 ; INPUTS:
                                07CE  1716 ;
                                07CE  1717 ; R3  = IRP (if called from FD~ routine) or 0 (otherwise)
                                07CE  1718 ; R9  = 2 bytes of mask and attributes
                                07CE  1719 ; R10 = Mask of characters to enable/disable
                                07CE  1720 ;
                                07CE  1721 ; OUTPUTS:
                                07CE  1722 ;
                                07CE  1723 ; R0 - status
                                07CE  1724 ; R1 - destroyed
                                07CE  1725 ; R2 - one byte past formatted message
                                07CE  1726 ; R3 - IRP (if non-zero on entry)
                                07CE  1727 ;     - CTP (if zero on entry)
                                07CE  1728 ; R10,R11 - destroyed
                                07CE  1729 ;
                                07CE  1730 ;---
                                07CE  1731
                                07CE  1732 SETUP_OUTBAND:
                                07CE  1733          ;
                                07CE  1734          ; Loop through to find out how many bits are set
                                07CE  1735          ;
                                07CE  1736
                          50    D4  07CE  1737          CLRL    R0                      ; Count
                      52  5A    D0  07D0  1738          MOVL    R10,R2
                                07D3  1739 10$:
              51  52  1F  00    EA  07D3  1740          FFS     #0,#31,R2,R1            ; Find first bit set
                          06    13  07D8  1741          BEQL    20$                     ; If zero, then exit
                          50    D6  07DA  1742          INCL    R0                      ; increment
                                07DC  1743 ;***   PUSHL   R1                      ; Save char to use later?
                    F3 52  51   E4  07DC  1744          BBSC    R1,R2,10$               ; Clear bit and loop
                                07E0  1745 20$:
                  51  50  05    C5  07E0  1746          MULL3   #5,R0,R1                ; length of data ***
                      51  2C    C0  07E4  1747          ADDL2   #CTP$W_CH_PARAM,R1      ; header
                          53    D5  07E7  1748          TSTL    R3                      ; irp or not?
                          05    13  07E9  1749          BEQL    30$                     ; branch if not
                            014B  30  07EB  1750          BSBW    FDT_ALLOC_MESSAGE       ; Allocate a message
                          09    11  07EE  1751          BRB     35$                     ; skip
                            01??  30  07F0  1752 30$:    BSBW    ALLOC_CTP               ; get message buffer
                          24        E9  07F3  1753          BLBC    R0,60$                  ; exit if error
                      53  52    D0  07F6  1754          MOVL    R2,R3                   ; Set CTP address
                                07F9  1755 35$:
                                07F9  1756          ASSUME  CTP$B_CH_FLAGS EQ CTP$B_MSGTYPE+1
                          0B    9B  07F9  1757          MOVZBW  #CTP$C_MT_CHAR,-
                        2A A2       07FB  1758                  CTP$B_MSGTYPE(R2)       ; Set message type
                    52  2C A2    9E  07FD  1759          MOVAB   CTP$W_CH_PARAM(R2),R2   ; Address of data in message
                                0801  1760 40$:
              5B  5A  1F  00    EA  0801  1761          FFS     #0,#31,R10,R11          ; Find first bit set
                          0F    13  0806  1762          BEQL    50$                     ; If zero, then exit
                  82  0202  8F    B0  0808  1763          MOVW    #<CH$C_CTERMa8!-
                                0810  1764                  CH$C_CT_CHAR_ATT>,(R2)+ ; set characteristic selector
                      82  5B    90  080D  1765          MOVB    R11,(R2)+               ; Set data
                      82  59    B0  0810  1766          MOVW    R9,(R2)+                ; Initialize longword
                    EA 5A  5B   E4  0813  1767          BBSC    R11,R10,40$             ; clear bit and loop
```

```
                     0817  1768 50$:
                50   01   D0  0817  1769          MOVL    #1,R0                   ; Set success
                     05   081A  1770 60$:         RSB
```

```
                     081B  1772          .SBTTL  FETCH_OOB_DATA - fetch data from OOB list
                     081B  1773 ;++
                     081B  1774 ;       FETCH_OOB_DATA - fetch data from OOB list
                     081B  1775 ;
                     081B  1776 ; Functional description:
                     081B  1777 ;
                     081B  1778 ;       Run through OOB list and build 3 masks - one for each
                     081B  1779 ;       type of OOB enable.
                     081B  1780 ;
                     081B  1781 ; Inputs:
                     081B  1782 ;
                     081B  1783 ;       R11 = points to 3 long word region, uses AST_ offsets
                     081B  1784 ;
                     081B  1785 ; Outputs:
                     081B  1786 ;
                     081B  1787 ;       AST_INCLUDE (R11)
                     081B  1788 ;       AST_EXCLUDE (R11)
                     081B  1789 ;       AST_ABORT (R11)
                     081B  1790 ;
                     081B  1791 ;       R0,R1,R2 destroyed
                     081B  1792 ;--
                     081B  1793 FETCH_OOB_DATA:
                     081B  1794
        6B   7C      081B  1795          CLRQ    (R11)                       ; clear
      08 AB  D4      081D  1796          CLRL    8(R11)                      ; clear
57  009C C5  9E      0820  1797          MOVAB   UCB$L_TL_BANDQUE(R5),R7     ; Address of the out of band list
                     0825  1798          DSBINT  UCB$B_DIPL(R5)              ; Interlock Queue access
                     082C  1799 10$:
    52   67  D0      082C  1800          MOVL    (R7),R2                     ; get next TAST block
         2D  13      082F  1801          BEQL    100$                        ; done
52    E4 A2  9E      0831  1802          MOVAB   -TAST$L_FLINK(R2),R2        ; get base of block
50    2D A2  9A      0835  1803          MOVZBL  TAST$B_CTRL(R2),R0          ; get control byte
1B 50    04  E0      0839  1804          BBS     #TAST$V_LOST,R0,50$         ; flagged for delete, ignore
51    30 A2  D0      083D  1805          MOVL    TAST$L_MASK(R2),R1          ; save mask
0F 50    05  E0      0841  1806          BBS     #TAST$V_ABORT,R0,40$        ; branch if abort
06 50    01  E0      0845  1807          BBS     #TAST$V_INCLUDE,R0,30$      ; branch if include
04 AB    51  C8      0849  1808          BISL    R1,AST_EXCLUDE(R11)         ; set exclude
         09  11      084D  1809          BRB     50$                         ; continue
    6B   51  C8      084F  1810 30$:     BISL    R1,AST_INCLUDE(R11)         ; set include
         04  11      0852  1811          BRB     50$                         ; continue
08 AB    51  C8      0854  1812 40$:     BISL    R1,AST_ABORT(R11)           ; add to abort flags
                     0858  1813
57    1C A2  DE      0858  1814 50$:     MOVAL   TAST$L_FLINK(R2),R7         ; point at flink
         CE  11      085C  1815          BRB     10$                         ; loop
                     085E  1816
                     085E  1817 100$:    ENBINT                              ; enable interrupts
              05     0861  1818          RSB
                     0862  1819
```

```
                     0862  1821          .SBTTL  CT_SENSEMODE, Function Decision Routine for SENSEMODE/SENSECHAR
                     0862  1822 ;++
                     0862  1823 ; CT_SENSEMODE, Function Decision Routine for SENSEMODE/SENSECHAR Functions
                     0862  1824 ;
                     0862  1825 ; Functional description:
                     0862  1826 ;
                     0862  1827 ;       This routine is called by the SYS$QIO service to dispatch a SENSEMODE
                     0862  1828 ;       or SENSECHAR I/O request.
                     0862  1829 ;
                     0862  1830 ;       The QIO parameters for terminal SENSEMODE/SENSECHAR are:
                     0862  1831 ;
                     0862  1832 ;       P1 = address of 8 or 12 byte characteristics buffer
                     0862  1833 ;       P2 = 0, 8 or 12
                     0862  1834 ;
                     0862  1835 ;       The buffer is validated for access, the process's quota checked and
                     0862  1836 ;       decremented, a message block is allocated, the address of the message
                     0862  1837 ;       block is stored in the IRP, and the IRP is queued to the ACP for
                     0862  1838 ;       delivery to the remote system.
                     0862  1839 ;
                     0862  1840 ; Inputs:
                     0862  1841 ;
                     0862  1842 ;       R0-R2 = scratch registers
                     0862  1843 ;       R3 = address of the IRP (I/O request packet)
                     0862  1844 ;       R4 = address of the PCB (process control block)
                     0862  1845 ;       R5 = address of the UCB (unit control block)
                     0862  1846 ;       R6 = address of the CCB (channel control block)
                     0862  1847 ;       R7 = bit number of the I/O function code
                     0862  1848 ;       R8 = address of the FDT table entry for this routine
                     0862  1849 ;       R9-R11 = scratch registers
                     0862  1850 ;       AP = address of the 1st function dependent QIO parameter
                     0862  1851 ;
                     0862  1852 ; Outputs:
                     0862  1853 ;
                     0862  1854 ;       IRP$L_SVAPTE(R3) = address of message buffer
                     0862  1855 ;       IRP$W_BOFF(R3) = size of message buffer
                     0862  1856 ;       IRP$L_MEDIA(R3) = address of user characteristics buffer
                     0862  1857 ;       IRP$W_BCNT(R3) = size of user characteristics buffer, 8
                     0862  1858 ;
                     0862  1859 ;       The routine preserves all registers except R0-R2, and R9-R11
                     0862  1860 ;--
                     0862  1861 CT_SENSEMODE:                                ; SENSEMODE/SENSECHAR FDT routine
                     0862  1862
          42 A3      B4  0862  1863          CLRW    IRP$W_CT_POST(R3)       ; Clear POSTed flag
           0478      30  0865  1864          BSBW    CT_WRITE_WRTCTP         ; See if flush must happen
                     0868  1865
     59    20 A3      3C  0868  1866          MOVZWL  IRP$W_FUNC(R3),R9       ; Fetch function code
   08 59     07      E1  086C  1867          BBC     #IO$V_RD_MODEM,R9,2$    ; skip if not read modem
50     0334 8F      3C  0870  1868          MOVZWL  #SS$_DEVREQERR, R0      ; Return as device request error
           00B0      31  0875  1869          BRW     FDT_ABORT               ; with an error not success
                     0878  1870 2$:
     51    6C      D0  0878  1871          MOVL    P1(AP),R1               ; Get address of characteristics buffer
           007D      30  087B  1872          BSBW    FDT_CHARSIZE            ; Size of chars buffer (return in R2)
     50    0C      3C  087E  1873          MOVZWL  #SS$_ACCVIO,R0          ; Assume access violation
                     0881  1874          IFWRT   R2,(R1),10$             ; Buffer accessible?
           009E      31  0887  1875          BRW     FDT_ABORT               ; Branch if not
                     088A  1876 10$:
   08 59     0E      E1  088A  1877          BBC     #IO$V_BRDCST,R9,12$     ; Branch if not brdcst bit request
```

```
                61   00A8 C5    7D   088E  1878          MOVQ    UCB$Q_TL_BRKTHRU(R5),(R1) ; read bits (no remoting of this?)
                       007F     31   0893  1879          BRW     FDT_FINISHIOC_OK          ; Complete I/O
                                     0896  1880 12$:
                     38 A3      51   D0   0896  1881          MOVL    R1,IRP$L_MEDIA(R3)        ; Save address in packet
                     32 A3      52   B0   089A  1882          MOVW    R2,IRP$W_BCNT(R3)         ; Set size in packet
                     2A A3      02   A8   089E  1883          BISW    #IRP$M_FUNC,IRP$W_STS(R3) ; Set READ type function
                                     08A2  1884          ;
                                     08A2  1885          ; For sense type-ahead, always use CTERM
                                     08A2  1886          ;
                     0C 59      06   E0   08A2  1887          BBS     #IO$V_TYPEAHDCNT,R9,15$ ; Branch if sense type-ahead
                                     08A6  1888          ;
                                     08A6  1889          ; Send CTERM or VAX message
                                     08A6  1890          ;
                           0F   E1   08A6  1891          BBC     #CTP$C_MT_VMSQIO,-
                     1C 0108 C5       08A8  1892                  UCB$L_CT_LEGALMSG(R5),20$ ; Branch if not VMSQIO support
                          F751'  30   08AC  1893          BSBW    CT_VMS_SENSEMODE          ; Format VAX message
                           0465  31   08AF  1894          BRW     CT_NETMSGSENDX            ; Send message and exit
                                     08B2  1895 15$:
                             61  7C   08B2  1896          CLRQ    (R1)                      ; Clear output buffer
                          51   2C   3C   08B4  1897          MOVZWL  #CTP$C_CI_LEN,R1          ; Set size of message buffer
                           007F  30   08B7  1898          BSBW    FDT_ALLOC_MESSAGE         ; Allocate the message buffer
                             0C  9B   08BA  1899          MOVZBW  #CTP$C_MT_CHECK_INP,-
                        2A A2        08BC  1900                  CTP$B_MSGTYPE(R2)
                     40 A3      0D   90   08BE  1901          MOVB    #CTP$C_MT_INP_COUNT, -
                                     08C2  1902                  IRP$B_CT_RESPTYPE(R3)     ; Set expected response type (input count)
                                     08C2  1903
                     52   2C A2      9E   08C2  1904          MOVAB   CTP$B_CI_FLAGS+1(R2),R2 ; Set end of message
                             1F      11   08C6  1905          BRB     30$
                                     08C8  1906 20$:
                                     08C8  1907          ;
                                     08C8  1908          ; CTP$AB_SENSEBUF is a fixed structure used to sense all characteristics
                                     08C8  1909          ; relevant to VMS.
                                     08C8  1910          ;
                           002E'8F   3C   08C8  1911          MOVZWL  #CTP$K_SENSEBUF+4+ -
                                51   08CC  1912                  CTP$B_MSGTYPE,R1          ; Set size of message buffer
                                     08CD  1913
                                0069 30   08CD  1914          BSBW    FDT_ALLOC_MESSAGE         ; Allocate the message buffer
                             40 A3   0B   90   08D0  1915          MOVB    #CTP$C_MT_CHAR, -
                                     08D4  1916                  IRP$B_CT_RESPTYPE(R3)     ; Set expected response type
                                     08D4  1917
                                   3C   BB   08D4  1918          PUSHR   #^M<R2,R3,R4,R5>          ; Save registers
                           0000'8F   28   08D6  1919          MOVC3   #CTP$K_SENSEBUF,-
                     2A A2 0000'CF       08DA  1920                  W^CTP$AB_SENSEBUF, -
                                     08DF  1921                  CTP$B_MSGTYPE(R2)         ; Copy data
                           51   53   D0   08DF  1922          MOVL    R3,R1                     ; Save adr beyond data
                                   3C   BA   08E2  1923          POPR    #^M<R2,R3,R4,R5>          ; Restore the registers
                           52   51   D0   08E4  1924          MOVL    R1,R2                     ; Pointer beyond data in message
                                     08E7  1925 30$:
                                0420 31   08E7  1926          BRW     CT_NETMSGSENDX            ; Send the message and exit service
```

```
                        08EA  1928          .SBTTL  FDT support routines
                        08EA  1929 ;++
                        08EA  1930 ; FDT_CHARSIZE
                        08EA  1931 ;
                        08EA  1932 ; INPUT:
                        08EA  1933 ;       P2(AP)
                        08EA  1934 ;
                        08EA  1935 ; OUTPUT:
                        08EA  1936 ;       R2 = 8 or 12 for size of characteristics buffer
                        08EA  1937 ;
                        08EA  1936 ; ABORT with status = SS$_BADPARAM if P2(AP) is not 0, 8, 12.
                        08EA  1939 ;
                        08EA  1940 ;--
                        08EA  1941
                        08EA  1942 GET_PARAMS:
                        08EA  1943
    51    6C    D0      08EA  1944          MOVL    P1(AP),R1               ; Get address of characteristics
          0C    10      08ED  1945          BSBB    FDT_CHARSIZE            ; Obtain the size of the char buffer
    50    0C    3C      08EF  1946          MOVZWL  #SS$_ACCVIO,R0          ; Assume access violation
                        08F2  1947          IFNORD  R2,(R1),10$             ; Characteristics accessible?
                05      08F8  1948          RSB                             ; return
                        08F9  1949 10$:
          2D    11      08F9  1950          BRB     FDT_ABORT               ; error
                        08FB  1951
                        08FB  1952 FDT_CHARSIZE:
  52   04 AC    D0      08FB  1953          MOVL    P2(AP), R2      ; Size of characters buffer
          0D    13      08FF  1954          BEQL    10$             ; Zero is for 8
     08   52    D1      0901  1955          CMPL    R2, #8          ; 8 is allowed
          0B    13      0904  1956          BEQL    20$             ; Ok
          0A    1F      0906  1957          BLSSU   30$             ; Less is no good
     0C   52    D1      0908  1958          CMPL    R2, #12         ; Must be 12 and nothing else
          05    12      090B  1959          BNEQ    30$             ; No good
                05      090D  1960          RSB                     ; Ok
    52    08    D0      090E  1961 10$:     MOVL    #8, R2          ; Use 8 if zero
                05      0911  1962 20$:     RSB
                        0912  1963
    50    14    3C      0912  1964 30$:     MOVZWL  #SS$_BADPARAM, R0       ; Abort qio with an error
                        0915  1965
                        0915  1966 ;
                        0915  1967 ; Finish I/O, clear R1
                        0915  1968 ;
                        0915  1969 FDT_FINISHIOC_OK:
    50    01    3C      0915  1970          MOVZWL  #SS$_NORMAL,R0          ; Set status OK
                        0918  1971 FDT_FINISHIOC:
          50    DD      0918  1972          PUSHL   R0                      ; save
          50    D4      091A  1973          CLRL    R0                      ; set position
       01CE     30      091C  1974          BSBW    CHECK_POST_IO           ; check for error, set posted
          50 8ED0      091F  1975          POPL    R0                      ; restore
00000000'GF    17      0922  1976          JMP     G^EXE$FINISHIOC         ; Complete I/O request
                        0928  1977 ;
                        0928  1978 ; Error processing - abort I/O request
                        0928  1979 ;
                        0928  1980
                        0928  1981 FDT_ABORT:                           ;
          50    DD      0928  1982          PUSHL   R0                      ; save
    50    01    9A      092A  1983          MOVZBL  #1,R0                   ; set position
       01BD     30      092D  1984          BSBW    CHECK_POST_IO           ; check for error, set posted
```

```
                50 8ED0  0930  1985          POPL    R0                      ; restore
00000000'GF      17  0933  1986          JMP     G^EXE$ABORTIO           ;
                     0939  1987
```

```
                     0939  1989          .SBTTL  FDT_ALLOC_MESSAGE, Allocate a message buffer
                     0939  1990 ;++
                     0939  1991 ; FDT_ALLOC_MESSAGE, Allocate a message buffer to send to remote process
                     0939  1992 ; SET_MSGHDR,    Setup a message header for broadcast
                     0939  1993 ;
                     0939  1994 ; Functional description:
                     0939  1995 ;
                     0939  1996 ;       This routine checks that the process has sufficient buffered I/O
                     0939  1997 ;       byte count quota for the message buffer, and then allocates the
                     0939  1998 ;       buffer from non-paged pool.  The process's buffered I/O byte count
                     0939  1999 ;       quota is decreased by the size of the allocated buffer and the
                     0939  2000 ;       message header information is stored.
                     0939  2001 ;
                     0939  2002 ; Inputs:
                     0939  2003 ;
                     0939  2004 ;       R1 = size of message required
                     0939  2005 ;       R3 = address of IRP
                     0939  2006 ;       R4 = address of PCB
                     0939  2007 ;
                     0939  2008 ; Outputs:
                     0939  2009 ;
                     0939  2010 ;       R1 = size of buffer
                     0939  2011 ;       R2 = address of buffer
                     0939  2012 ;
                     0939  2013 ;       IRP$B_CT_RESPTYPE = response type cleared
                     0939  2014 ;       IRP$L_SVAPTE(R3) = address of buffer
                     0939  2015 ;       IRP$W_BOFF(R3) = size of buffer
                     0939  2016 ;
                     0939  2017 ;       CTP$B_TYPE(R2)          = Block type
                     0939  2018 ;       CTP$W_SIZE(R2)          = size of message buffer
                     0939  2019 ;       CTP$W_MSGSIZE(R2)       = message size, zeroed
                     0939  2020 ;       CTP$B_PRO_MSGTYPE(R2)   = PRO$C_DATA
                     0939  2021 ;       CTP$B_CT_RESPTYPE(R2)   = response type, zeroed
                     0939  2022 ;       CTP$L_MSGDAT(R2)        = address of data
                     0939  2023 ;       CTP$L_USRBFR(R2)        = zeroed
                     0939  2024 ;
                     0939  2025 ;       If process does not have sufficient quota, the I/O request
                     0939  2026 ;       is aborted.
                     0939  2027 ;--
                     0939  2028
                     0939  2029 FDT_ALLOC_MESSAGE::                       ; Allocate message buffer
                     0939  2030
          53    DD   0939  2031          PUSHL   R3                      ; Save packet address
00000000'GF    16   093B  2032          JSB     G^EXE$BUFFRQUOTA        ; Check quota
       35 50   E9   0941  2033          BLBC    R0,ALLOC_ERR            ; Branch if error
                     0944  2034 ;
                     0944  2035 ; Allocate the message buffer
                     0944  2036 ;
00000000'GF    16   0944  2037 10$:     JSB     G^EXE$ALLOCBUF          ; Allocate the buffer
       2C 50   E9   094A  2038          BLBC    R0,ALLOC_ERR            ; Branch if error
          53 8ED0   094D  2039          POPL    R3                      ; Restore packet address
                     0950  2040 ;
                     0950  2041 ; Adjust process's quota
                     0950  2042 ;
50    0080 C4   D0   0950  2043          MOVL    PCB$L_JIB(R4),R0        ; Get Job Information Block address
  20 A0   51   C2   0955  2044          SUBL    R1,JIB$L_BYTCNT(R0)     ; Adjust buffered I/O byte count quota
  30 A3   51   B0   0959  2045          MOVW    R1,IRP$W_BOFF(R3)       ; Save buffer size as quota
```

```
                     095D  2046 ;
                     095D  2047 ; Set up CTP with associated IRP data
                     095D  2048 ;
2C A3   52   D0      095D  2049         MOVL    R2,IRP$L_SVAPTE(R3)     ; Save buffer address in packet
                     0961  2050         ASSUME IRP$B_CT_RESPTYPE+1 EQ IRP$B_CT_CANCEL
                     0961  2051         ASSUME IRP$B_CT_CANCEL+1   EQ IRP$W_CT_POST
     40 A3   D4      0961  2052         CLRL    IRP$B_CT_RESPTYPE(R3)   ; Set no response type, cancel or post
                     0964  2053 ;
                     0964  2054 ; Store message header information
                     0964  2055 ;
                     0964  2056 ;       R0      = Clobbered
                     0964  2057 ;       R1      = Buffer size
                     0964  2058 ;       R2      = Buffer address
                     0964  2059 ;
                     0964  2060
                     0964  2061 SET_MSGHDR:
                     0964  2062
                     0964  2063         ASSUME  CTP$B_PRO_MSGTYPE+1 EQ CTP$B_PRO_FILL
                     0964  2064         ASSUME  CTP$B_PRO_MSGTYPE+2 EQ CTP$W_MSGSIZE
                     0964  2065
        09   9A      0964  2066         MOVZBL  #PRO$C_DATA,-
     26 A2           0966  2067                 CTP$B_PRO_MSGTYPE(R2)   ; Assume data type, zero fill & msgsize
08 A2   51   B0      0968  2068         MOVW    R1,CTP$W_SIZE(R2)       ; Save buffer size in message
        13   90      096C  2069         MOVB    #DYN$C_BUFIO,-          ; Set block type
     0A A2           096E  2070                 CTP$B_TYPE(R2)
     26 A2   9E      0970  2071         MOVAB   CTP$B_PRO_MSGTYPE(R2),- ; Set address of data
     0C A2           0973  2072                 CTP$L_MSGDAT(R2)
     10 A2   D4      0975  2073         CLRL    CTP$L_USRBFR(R2)        ; Set user buffer address
             05      0978  2074         RSB                             ;
                     0979  2075
                     0979  2076 ALLOC_ERR:
                     0979  2077
   53 8ED0           0979  2078         POPL    R3                      ; Restore R3
FFA9   31            097C  2079         BRW     FDT_ABORT               ; Abort the IO
```

```
                        097F  2081 .SBTTL  ALLOC_CTP - allocate a CTP
                        097F  2082
                        097F  2083 ALLOC_CTP:
                        097F  2084 ;
                        097F  2085 ; Allocates a CTP and does not charge quota.
                        097F  2086 ;
                        097F  2087 ;
00000000'GF   16        097F  2088         JSB     G^EXE$ALONONPAGED       ; allocate CTP
      06 50   E9        0985  2089         BLBC    R0,10$                  ; Branch on error
        FFD9  30        0988  2090         BSBW    SET_MSGHDR              ; Set up message header
   50   01   D0        098B  2091         MOVL    #1,R0                   ; success
                        098E  2092 10$:
              05        098E  2093         RSB                             ; return
                        098F  2094
```

```
                        2096          .SBTTL  CT_INTERRUPT Interrupt handler
                        2097 ;++
                        2098 ; CT_INTERRUPT, I/O completion interrupt handler
                        2099 ;
                        2100 ; Functional description:
                        2101 ;
                        2102 ;       This routine handles an I/O completion "interrupt" from the ACP.
                        2103 ;       The I/O status and data is obtained from the response packet from
                        2104 ;       the remote terminal handler process, and the I/O request is completed.
                        2105 ;
                        2106 ; Inputs:
                        2107 ;
                        2108 ;       R3 = address of the IRP
                        2109 ;       R5 = address of UCB
                        2110 ;       IRP$L_SVAPTE(R3) = address of response message
                        2111 ;
                        2112 ;       ipl = ipl$_iopost
                        2113 ;
                        2114 ; Legal message types are:
                        2115 ;
                        2116 ; 3  - read data
                        2117 ; 8  - write complete
                        2118 ; 11 - characteristics
                        2119 ; 13 - input count
                        2120 ;
                        2121 ; Outputs:
                        2122 ;
                        2123 ;       I/O status copied to IRP$L_IOST and I/O request posted.
                        2124 ;
                        2125 ;       R0-R2 are scratch.
                        2126 ;
                        2127 ;--
                        2128 CT_INTERRUPT:                            ; I/O completion interrupt handler
     56   DD            2129          PUSHL   R6                      ; Save R6
0200 8F   A8            2130          BISW    #IRP$M_TERMIO,-
  2A A3                 2131                  IRP$W_STS(R3)           ; Set terminal I/O completion
56   38 A3   D0         2132          MOVL    IRP$L_MEDIA(R3),R6      ; Save media (same as IOST1 location)
     01   B0            2133          MOVW    #SS$_NORMAL,-
  38 A3                 2134                  IRP$L_IOST1(R3)         ; Assume success
52   2C A3   D0         2135          MOVL    IRP$L_SVAPTE(R3),R2     ; Get address of message
     26   C3            2136          SUBL3   #CTP$B_PRO_MSGTYPE,-
51   62                 2137                  (R2),R1                 ; Point to message
                        2138
50   2A A1   9A         2139          MOVZBL  CTP$B_MSGTYPE(R1),R0    ; fetch message type
     50   03   91        2140          CMPB    #CTP$C_MT_READ_DATA,R0  ; is it a read?
          20   13        2141          BEQL    POST_READ               ; Branch if yes
     50   08   91        2142          CMPB    #CTP$C_MT_WRITE_COM,R0  ; is it a write?
          03   12        2143          BNEQ    10$
        00DD   31        2144          BRW     POST_WRITE              ; Branch if yes
                        2145 10$:
     50   0B   91        2146          CMPB    #CTP$C_MT_CHAR,R0       ; is it a read char?
          03   12        2147          BNEQ    15$                     ; nope
        0082   31        2148          BRW     POST_SENSE              ; Branch if yes
                        2149 15$:
     50   0D   91        2150          CMPB    #CTP$C_MT_INP_COUNT,R0  ; Input count?
          03   12        2151          BNEQ    20$
        00E9   31        2152          BRW     POST_SENSE_TYPEAHD      ; post send type ahead
```

```
                        09C8  2153
                        09C8  2154 20$:     MINOR_ERROR                         ; increment error count
                56 8ED0 09CC  2155          popl    R6                          ;
                     05 09CF  2156          RSB                                 ; Is this a minor error?
                        09D0  2157 ;
                        09D0  2158 ; Set up buffer to post READ
                        09D0  2159 ;
                        09D0  2160 POST_READ:
                        09D0  2161
                02   AA 09D0  2162          BICW    #FLG$M_CTRLO,-
           00DC C5      09D2  2163                  UCB$W_CT_FLAGS(R5)          ; Set no control O
        62   32 A1   9E 09D5  2164          MOVAB   CTP$T_RD_DATA(R1),(R2)      ; Set address of data
     04 A2   56   D0 09D9  2165          MOVL    R6,4(R2)                    ; Set address of user buffer
                        09DD  2166 ;
                        09DD  2167 ; Fill in IOSB status
                        09DD  2168 ;
     50   3C A3   3C 09DD  2169          MOVZWL  IRP$L_IOST2(R3),R0          ; Fetch net message data length
        38 A3   7C 09E1  2170          CLRQ    IRP$L_IOST1(R3)             ; clear status
           50   0C   C2 09E4  2171          SUBL2   #<CTP$T_RD_DATA- -
                        09E7  2172                  CTP$B_PRO_MSGTYPE>,R0       ; subtract out header
     32 A3   50   B1 09E7  2173          CMPW    R0,IRP$W_BCNT(R3)           ; Size of data greater than user buffer?
                06   1A 09EB  2174          BGTRU   10$                         ; If gtru yes - leave user's size
     32 A3   50   B0 09ED  2175          MOVW    R0,IRP$W_BCNT(R3)           ; Else, set size to actual data size
                23   13 09F1  2176          BEQL    20$                         ; If no data, terminator data not needed
                        09F3  2177 ;
                        09F3  2178 ; Set terminator data into IOSB
                        09F3  2179 ;
                        09F3  2180 10$:
     56   30 A1   3C 09F3  2181          MOVZWL  CTP$W_RD_TERM_POS(R1),R6 ; Fetch offset to terminator
        3A A3   56   B0 09F7  2182          MOVW    R6,IRP$L_IOST1+2(R3)    ; Set offset to terminator
  3E A3   50   56   A3 09FB  2183          SUBW3   R6,R0,IRP$L_IOST1+6(R3) ; Size of terminator
        50   62   D0 0A00  2184          MOVL    (R2),R0                     ; Fetch address of data
     3C A3   6046   9B 0A03  2185          MOVZBW  (R0)[R6],IRP$L_IOST1+4(R3) ; Terminator character
                        0A08  2186 ;
                        0A08  2187 ; Skip final fill in if not extended mode read
                        0A08  2188 ;
                0F   E1 0A08  2189          BBC     #IO$V_EXTEND,-
           09 20 A3     0A0A  2190                  IRP$W_FUNC(R3),20$
        3D A3   01   8E 0A0D  2191          MNEGB   #1,IRP$L_IOST1+5(R3)    ; set unused byte to -1
             2F A1   90 0A11  2192          MOVB    CTP$B_RD_CURS_POS(R1),-
                3F A3   0A14  2193                  IRP$L_IOST1+7(R3)           ; Set cursor position from EOL
                        0A16  2194 ;
                        0A16  2195 ; Map TSA read status to VMS read status
                        0A16  2196 ;
                        0A16  2197 20$:
             04   00   EF 0A16  2198          EXTZV   #0,#4,-
        50   2B A1      0A19  2199                  CTP$B_RD_FLAGS(R1),R0   ; fetch status bits from read data flags
                50   04 91 0A1C  2200          CMPB    #CTP$M_RD_INPFULL,R0    ; input buffer was full?
                   03   12 0A1F  2201          BNEQ    30$                     ; if no, branch
             3C A3   D4 0A21  2202          CLRL    IRP$L_IOST1+4(R3)       ; Clear terminator data
  38 A3   F64F CF40   B0 0A24  2203 30$:     MOVW    STAT_TABLE[R0],-
                        0A2B  2204                  IRP$L_IOST1(R3)         ; Get VMS status
                38 A3   B1 0A2B  2205          CMPW    IRP$L_IOST1(R3),-
             0611 8F    0A2E  2206                  #SS$_CONTROLY           ; CONTROL Y?
                   0C   12 0A31  2207          BNEQ    50$                     ; no, continue
                   07   E1 0A33  2208          BBC     #FLG$V_CTRLC,-
        06 00DC C5      0A35  2209                  UCB$W_CT_FLAGS(R5),50$  ; branch if ^C not just delivered
```

```
                     0651 8F    B0   0A39  2210          MOVW    #SS$_CONTROLC,-
                       38 A3          0A3D  2211                  IRP$L_IOST1(R3)         ; set VMS status to ^C
                        0087    31    0A3F  2212 50$:     BRW     POST                    ;
                                      0A42  2213
                                      0A42  2214 ;
                                      0A42  2215 ; Set up buffer to post SENSEMODE/CHAR
                                      0A42  2216 ;
                                      0A42  2217 POST_SENSE:                            ;
                                      0A42  2218
                  62    2C A1    9E   0A42  2219          MOVAB   CTP$W_CH_PARAM(R1),(R2) ; Set address of data
                  04 A2    56    D0   0A46  2220          MOVL    R6,4(R2)                ; Set address of user data
                                      0A4A  2221
                     0FD6 8F    BB   0A4A  2222          PUSHR   #^M<R1,R2,R4,R6,R7,R8,R9,R10,R11>        ; save
                     52    51   D0   0A4E  2223          MOVL    R1,R2                   ; Set address of of CTP
                  59    6C A3    9E   0A51  2224          MOVAB   IRP$L_FPC(R3),R9        ; Set address of 12 byte buffer
                  5A    38 A3    9E   0A55  2225          MOVAB   IRP$L_IOST1(R3),R10     ; Set address of 8 byte buffer
                           69    7C   0A59  2226          CLRQ    (R9)                    ; clear
                        08 A9    D4   0A5B  2227          CLRL    8(R9)                   ; clear
                           6A    7C   0A5E  2228          CLRQ    (R10)                   ; clear
                                      0A60  2229
                         F59D'   30   0A60  2230          BSBW    CT_POST_SENSE           ; Map list into qio data
                                      0A63  2231
                     0FD6 8F    BA   0A63  2232          POPR    #^M<R1,R2,R4,R6,R7,R8,R9,R10,R11> ; Restore
                                      0A67  2233
                       6C A3    7D   0A67  2234          MOVQ    IRP$L_FPC(R3),-
                       2C A1         0A6A  2235                  CTP$W_CH_PARAM(R1)      ; Move data into place
                       74 A3    D0   0A6C  2236          MOVL    IRP$L_FR4(R3),-
                       34 A1         0A6F  2237                  CTP$W_CH_PARAM+8(R1)    ; Move data into place
                                      0A71  2238 ;
                                      0A71  2239 ; Set bits that can't be read through CTERM
                                      0A71  2240 ;
               50   44 A5    FF28DFFA 8F   CB   0A71  2241          BICL3   #REMOTE_1, -
                                      0A7A  2242                  UCB$L_DEVDEPEND(R5),R0  ; Clear all bits that can be read
                        30 A1    50   C8   0A7A  2243          BISL    R0,CTP$W_CH_PARAM+4(R1) ; Set bits that can't in return data
                     50    48 A5    02   CB   0A7E  2244          BICL3   #REMOTE_2, -
                                      0A83  2245                  UCB$L_DEVDEPND2(R5),R0  ; Clear all bits that can be read
                        34 A1    50   C8   0A83  2246          BISL    R0,CTP$W_CH_PARAM+8(R1) ; Set bits that can't in return data
                           2C A1    D0   0A87  2247          MOVL    CTP$W_CH_PARAM(R1),-
                           40 A5         0A8A  2248                  UCB$B_DEVCLASS(R5)      ; store class, type and bufsiz
                           30 A1    7D   0A8C  2249          MOVQ    CTP$W_CH_PARAM+4(R1),-
                           44 A5         0A8F  2250                  UCB$L_DEVDEPEND(R5)     ; store in UCB
                              46    10   0A91  2251          BSBB    SENSE_SPAWN             ; sense bits for DCL
                              34    11   0A93  2252          BRB     POST
                                      0A95  2253
                                      0A95  2254 POST_WRITE:
                                      0A95  2255
                              00    E1   0A95  2256          BBC     #CTP$V_WC_DISCARD,-
                        06 2B A1         0A97  2257                  CTP$B_WC_FLAGS(R1),10$  ; branch if not ^Oed
                         0609 8F    B0   0A9A  2258          MOVW    #SS$_CONTROLO,-
                           38 A3         0A9E  2259                  IRP$L_IOST1(R3)         ; Set ^O status
                                      0AA0  2260 10$:
                           32 A3    B0   0AA0  2261          MOVW    IRP$W_BCNT(R3),-
                           3A A3         0AA3  2262                  IRP$L_IOST1+2(R3)       ; Set byte count ?
                           2C A1    90   0AA5  2263          MOVB    CTP$W_WC_HORPOS(R1),-
                           3E A3         0AA8  2264                  IRP$L_IOST1+6(R3)       ; horizontal position
                           2E A1    90   0AAA  2265          MOVB    CTP$W_WC_VERPOS(R1),-
                           3F A3         0AAD  2266                  IRP$L_IOST1+7(R3)       ; vertical position
```

```
                    18   11  0AAF  2267          BRB     POST
                                0AB1  2268
                                0AB1  2269 POST_SENSE_TYPEAHD:
                                0AB1  2270
              50    3C A3   3C  0AB1  2271          MOVZWL  IRP$L_IOST2(R3),R0      ; Fetch net message data length
                 38 A3    7C  0AB5  2272          CLRQ    IRP$L_IOST1(R3)         ; IOSB *** ?
              38 A3    01   B0  0AB8  2273          MOVW    #SS$_NORMAL,IRP$L_IOST1(R3) ; IOSB
              62    2C A1   9E  0ABC  2274          MOVAB   CTP$W_IC_COUNT(R1),(R2) ; Set address of data
              04 A2    56   D0  0AC0  2275          MOVL    R6,4(R2)                ; Set address of user buffer
        32 A3    50   06   A3  0AC4  2276          SUBW3   #CTP$C_IC_MSGLEN-2,R0,- ; normal msglen - 2 gives return length
                                0AC9  2277                  IRP$W_BCNT(R3)          ; note this works for VMS "longer" IC msg
                                0AC9  2278 POST:
                    56 8ED0     0AC9  2279          POPL    R6                      ; Restore R6
                                0ACC  2280          POST_IO                         ; post I/O
                 50    02   9A  0ACC               MOVZBL  #POST_COUNT,R0          ; set position
                    001B   30  0ACF               BSBW    CHECK_POST_IO           ; check for error, set posted
           00000000'GF    16  0AD2               JSB     G^COM$POST              ; post I/O
                          05  0AD8  2281          RSB
```

```
                        0AD9  2283          .SBTTL  SENSE_SPAWN  Sense for spawn
                        0AD9  2284
                        0AD9  2285 ;        Sense special characteristics bits for DCL spawn command.
                        0AD9  2286 ;        Return bits for ctrl/c ast, outofband ast and associated mailbox.
                        0AD9  2287 ;        These bits may be reused later and are not for customer application
                        0AD9  2288 ;        consumption.
                        0AD9  2289 ;
                        0AD9  2290 ; inputs:
                        0AD9  2291 ;        r1 -> CTP message
                        0AD9  2292 ;        R5 -> UCB
                        0AD9  2293
                        0AD9  2294 SENSE_SPAWN:
   50    34 A1    9E    0AD9  2295          MOVAB   CTP$W_CH_PARAM+8(R1),R0 ; Address of the characteristics
60    0200 8F    AA    0ADD  2296          BICW    #TT2$M_DCL_MAILBX,(R0)  ; Reset mailbox
      60 A5      D5    0AE2  2297          TSTL    UCB$L_AMB(R5)           ; Any associated mailbox?
         05      13    0AE5  2298          BEQL    10$                     ; No
60    0200 8F    A8    0AE7  2299          BISW    #TT2$M_DCL_MAILBX,(R0)  ; Yes, so set characteristic
                        0AEC  2300 10$:
                  05    0AEC  2301          RSB
```

```
                     0AED  2303 .SBTTL  CHECK_POST_IO, validate I/O to be posted
                     0AED  2304
                     0AED  2305 ;++
                     0AED  2306 ;
                     0AED  2307 ; Make paranoia check that I/O about to be posted has not been posted before.
                     0AED  2308 ; Bugcheck system if it has.
                     0AED  2309 ;
                     0AED  2310 ; Input:
                     0AED  2311 ;       R0 - post location
                     0AED  2312 ;       R3 - IRP
                     0AED  2313 ;
                     0AED  2314 ; Output:
                     0AED  2315 ;
                     0AED  2316 ;       IRP$W_CT_POST is set as "posted"
                     0AED  2317 ;
                     0AED  2318 ;--
                     0AED  2319
                     0AED  2320 CHECK_POST_IO:
                     0AED  2321
          42 A3   B5 0AED  2322         TSTW    IRP$W_CT_POST(R3)       ; Error if non-zero
             06   12 0AF0  2323         BNEQ    20$                     ; branch if problems
00 42 A3    50   E2 0AF2  2324         BBSS    R0,IRP$W_CT_POST(R3),10$; set posted flag
                 05 0AF7  2325 10$:     RSB
                     0AF8  2326
                     0AF8  2327 20$:    BUG_CHECK CTERM,FATAL           ; should never happen!
                 05  0AFC  2328         RSB                             ; never get here
                     0AFD  2329
```

```
                        0AFD  2331          .SBTTL  CT_CANCEL, Cancel I/O routine
                        0AFD  2332 ;++
                        0AFD  2333 ; CT_CANCEL, Cancels an I/O operation in progress
                        0AFD  2334 ;
                        0AFD  2335 ; Functional description:
                        0AFD  2336 ;
                        0AFD  2337 ;       This routine cancels any CTRL/C or CTRL/Y AST's that were
                        0AFD  2338 ;       requested by the cancelling process on the cancelling channel.
                        0AFD  2339 ;
                        0AFD  2340 ;       If there are no more references remaining to the device, the UCB
                        0AFD  2341 ;       is queued to the ACP to notify it that the device is no longer in
                        0AFD  2342 ;       use.  The ACP will then check that the reference count is still zero
                        0AFD  2343 ;       and remove the UCB from I/O database and deallocate it.
                        0AFD  2344 ;
                        0AFD  2345 ; Inputs:
                        0AFD  2346 ;
                        0AFD  2347 ;       R2 = negated value of the channel index number
                        0AFD  2348 ;       R3 = address of the current IRP (I/O request packet)
                        0AFD  2349 ;       R4 = address of the PCB (process control block) for the
                        0AFD  2350 ;                process canceling I/O
                        0AFD  2351 ;       R5 = address of the UCB (unit control block)
                        0AFD  2352 ;       R8 = Cancel reason code
                        0AFD  2353 ;
                        0AFD  2354 ;       IPL = driver fork IPL
                        0AFD  2355 ;
                        0AFD  2356 ; Outputs:
                        0AFD  2357 ;
                        0AFD  2358 ;       DEV$M_DMT is set in UCB$L_DEVCHAR to prevent a race if someone
                        0AFD  2359 ;       assigns and deassigns another channel to the UCB before the ACP
                        0AFD  2360 ;       dequeues the UCB.
                        0AFD  2361 ;
                        0AFD  2362 ;       The routine preserves all registers except R0-R3.
                        0AFD  2363 ;--
                        0AFD  2364 .ENABLE LOCAL_BLOCK
                        0AFD  2365
                        0AFD  2366 ASSUME  CAN$C_CANCEL EQ 0
                        0AFD  2367 ASSUME  CAN$C_DASSGN EQ 1
                        0AFD  2368
            0093    31  0AFD  2369 10$:    BRW     130$
            0087    31  0B00  2370 20$:    BRW     120$
                        0B03  2371
                        0B03  2372 CT_CANCEL:                              ; Cancel an I/O operation
                        0B03  2373
            01DA    30  0B03  2374          BSBW    CT_WRITE_WRTCTP        ; Flush write buffer
                        0B06  2375
        0EF0 8F     BB  0B06  2376          PUSHR   #^M<R4,R5,R6,R7,R9,R10,R11>     ; Save registers
             04     E1  0B0A  2377          BBC     #UCB$V_ONLINE,-        ; If clr unit offline - probably t mplate
        EE 64 A5        0B0C  2378                  UCB$W_STS(R5),10$      ;
           5C A5    B5  0B0F  2379          TSTW    UCB$W_REFC(R5)         ; Any more references to device?
              EC    13  0B12  2380          BEQL    20$                    ; Nope all done.
                        0B14  2381          ;
                        0B14  2382          ; Send UNREADs for all current reads in server.
                        0B14  2383          ;
            0585    30  0B14  2384          BSBW    CT_SEND_UNREAD         ; Cancel outstanding read IRPs
                        0B17  2385          ;
                        0B17  2386          ; Cancel all IRPs waiting in STALLQ that match this cancel
                        0B17  2387          ;
```

```
                     0B6B  2419 ;
                     0B6B  2420 ; Check if Out of band's were disabled
                     0B6B  2421 ;
59   0B   3C   0B6B  2422          MOVZWL  #CTP$M_CH_OO!CTP$M_CH_D,R9      ; Set up mask and cancel code
   FBFE   30   0B6E  2423          BSBW    SEND_OOB_MSG                    ; send out of band message
                     0B71  2424 ;
                     0B71  2425 ; Update UCB OOB masks
                     0B71  2426 ;
5E   0C   C2   0B71  2427          SUBL    #12,SP                          ; allocate 3 longwords
5B   5E   D0   0B74  2428          MOVL    SP,R11                          ; pointer
   FCA1   30   0B77  2429          BSBW    FETCH_OOB_DATA                  ; fetch data from OOB LIST
                     0B7A  2430          ;
                     0B7A  2431          ; Update UCB with new masks
                     0B7A  2432          ;
     6B   7D   0B7A  2433          MOVQ    AST_INCLUDE(R11),-
0128 C5            0B7C  2434                  UCB$L_CT_INCLUDE(R5)    ; set new OOB data
  08 AB   D0   0B7F  2435          MOVL    AST_ABORT(R11),-
0130 C5            0B82  2436                  UCB$L_CT_ABORT(R5)      ; and abort
5E   0C   C0   0B85  2437          ADDL    #12,SP                          ; reset stack
                     0B88  2438
     09   11   0B88  2439          BRB     130$
                     0B8A  2440 ;
                     0B8A  2441 ; Clean up the ucb after all references have gone.
                     0B8A  2442 ; An attempt is made to get an unbind through.  There is no
                     0B8A  2443 ; guarantee made about this message really being sent.
                     0B8A  2444 ;
```

```
                        0B8A  2445 120$:
50   03   9A  0B8A  2446          MOVZBL  #UNBIND$C_USER,R0       ; unbind reason code
   055C   30  0B8D  2447          BSBW    SEND_UNBIND             ; send it with common routine
   007B   30  0B90  2448          BSBW    CT_ABORTIRPS            ; Flush all irps from queue
                        0B93  2449                                  ; Insert UCB in ACP queue
                        0B93  2450 130$:
0EF0 8F    BA  0B93  2451          POPR    #^M<R4,R5,R6,R7,R9,R10,R11>     ; Restore registers
           05  0B97  2452          RSB                             ; Return
                        0B98  2453
                        0B98  2454 .DISABLE LOCAL_BLOCK
                        0B98  2455
                        0B98  2456 ;
                        0B98  2457 ; R1 - address of IRP list to run down
                        0B98  2458 ; R2 - channel number
                        0B98  2459 ; R4 - PCB
                        0B98  2460 ;
                        0B98  2461
                        0B98  2462 CANCEL_LIST:
                        0B98  2463
57   51   D0  0B98  2464          MOVL    R1,R7                   ; copy
56   52   D0  0B9B  2465          MOVL    R2,R6                   ; Make a copy of channel number
                        0B9E  2466 30$:
57   67   D0  0B9E  2467          MOVL    (R7),R7                 ; Get next IRP
51   57   D1  0BA1  2468          CMPL    R7,R1                   ; At list head?
     2F   13  0BA4  2469          BEQL    35$                     ; exit if yes
60 A4  0C A7  D1  0BA6  2470          CMPL    IRP$L_PID(R7),PCB$L_PID(R4) ; PID match?
     F1   12  0BAB  2471          BNEQ    30$                     ; branch if no
28 A7   56   B1  0BAD  2472          CMPW    R6,IRP$W_CHAN(R7)       ; channel match?
     EB   12  0BB1  2473          BNEQ    30$                     ; branch if no
41 A7   95  0BB3  2474          TSTB    IRP$B_CT_CANCEL(R7)     ; UNREAD send?
     E6   12  0BB6  2475          BNEQ    30$                     ; if so, skip post and wait for server
     53   DD  0BB8  2476          PUSHL   R3                      ; save
53   57   D0  0BBA  2477          MOVL    R7,R3                   ; save IRP
57   04 A7   D0  0BBD  2478          MOVL    4(R7),R7                ; move back one link
53   63   0F  0BC1  2479          REMQUE  (R3),R3                 ; remove from queue
                        0BC4  2480          POST_IO                 ; post I/O
50   03   9A  0BC4              MOVZBL  #POST_COUNT,R0          ; set position
   FF23   30  0BC7              BSBW    CHECK_POST_IO           ; check for error, set posted
00000000'GF   16  0BCA              JSB     G^COM$POST              ; post I/O
     53 8ED0  0BD0  2481          POPL    R3                      ; restore
     C9   11  0BD3  2482          BRB     30$                     ; loop
                        0BD5  2483 35$:
           05  0BD5  2484          RSB
```

```
                        0BD6  2486          .SBTTL  CT_HANGUP  - Perform hangup functions
                        0BD6  2487          .SBTTL  CT_ABORTIRPS - Abort irps outstanding
                        0BD6  2488 ;++
                        0BD6  2489 ; CT_HANGUP  Perform hangup functions
                        0BD6  2490 ; CT_ABORTIRPS
                        0BD6  2491 ;
                        0BD6  2492 ; Functional description:
                        0BD6  2493 ;
                        0BD6  2494 ;       Deliver any CTRL/Y AST's, specifying hang-up;
                        0BD6  2495 ;       deliver a hangup message to associated mailbox.
                        0BD6  2496 ;       Post any irps outstanding with abort.
                        0BD6  2497 ;       Set hangup status in device status.
                        0BD6  2498 ;       The ucb is passed on to the acp if there are no more
                        0BD6  2499 ;       channels open to it.
                        0BD6  2500 ;       HANGUP is called by net device errors and hangup operations
                        0BD6  2501 ;       from the line on the other end.
                        0BD6  2502 ;       ABORTIRPS is called on net device cancels and channel deassigns.
                        0BD6  2503 ;
                        0BD6  2504 ; Inputs:
                        0BD6  2505 ;
                        0BD6  2506 ;       R5 = address of UCB
                        0BD6  2507 ;
                        0BD6  2508 ;
                        0BD6  2509 ; Outputs:
                        0BD6  2510 ;
                        0BD6  2511 ;       Message or AST(s) delivered.
                        0BD6  2512 ;
                        0BD6  2513 ;--
                        0BD6  2514 CT_HANGUP:
    54  0090 C5   DE      0BD6  2515          MOVAL   UCB$L_TL_CTRLY(R5),R4   ; Get address of CTRL/Y AST list
        50   54   D0      0BDB  2516          MOVL    R4,R0                   ; Copy list address
                          0BDE  2517 10$:
        50   60   D0      0BDE  2518          MOVL    (R0),R0                 ; Get address of next entry
             08   13      0BE1  2519          BEQL    20$                     ; If eql none
        02CC 8F   3C      0BE3  2520          MOVZWL  #SS$_HANGUP,-           ; Insert new parameter for AST
          1C A0           0BE7  2521                  ACB$L_KAST+4(R0)        ;
             F3   11      0BE9  2522          BRB     10$                     ;
                          0BEB  2523 20$:
             56   DD      0BEB  2524          PUSHL   R6                      ; save (Necessary?)
    56   00A4 C5   D0      0BED  2525          MOVL    UCB$L_TL_CTLPID(R5),R6  ; Set controlling PID
     00000000'GF   16      0BF2  2526          JSB     G^COM$DELATTNASTP       ; Deliver the AST's
             56 8ED0      0BF8  2527          POPL    R6                      ; restore
        54   06   D0      0BFB  2528          MOVL    #MSG$_TRMHANGUP,R4      ; Set mailbox message type
     53   60 A5   D0      0BFE  2529          MOVL    UCB$L_AMB(R5),R3        ; Get associated mailbox address
             06   13      0C02  2530          BEQL    30$                     ; If eql none - forget it
     00000000'GF   16      0C04  2531          JSB     G^EXE$SNDEVMSG          ; Deliver notification to mailbox
                          0C0A  2532 30$:
             08   A8      0C0A  2533          BISW    #UCB$M_TT_HANGUP,-      ; Save hangup status
           68 A5           0C0C  2534                  UCB$W_DEVSTS(R5)        ;
                          0C0E  2535
```

```
                        0C0E  2537 ;
                        0C0E  2538 ;       Clean up the outstanding iirp read to network so it completes
                        0C0E  2539 ;       without calling driver again.  Post all outstanding irps with
                        0C0E  2540 ;       abort.
                        0C0E  2541 ;
                        0C0E  2542
                        0C0E  2543 CT_ABORTIRPS:
                        0C0E  2544
                        0C0E  2545 ;
                        0C0E  2546 ; Release TQE
                        0C0E  2547 ;
                        0C0E  2548         DSBINT  #IPL$_TIMER             ; Synch with timer queue ***
    50   00E4 C5   D0   0C14  2549         MOVL    UCB$L_CT_TQE(R5),R0     ; Get TQE address
              10   13   0C19  2550         BEQL    7$                      ; Branch if none
         00E4 C5   D4   0C1B  2551         CLRL    UCB$L_CT_TQE(R5)        ; Clear address
               02   88   0C1F  2552         BISB    #TQE$M_DELETE,-         ; Assume timer is busy,
             10 A0        0C21  2553                 TQE$L_FR3(R0)           ;  set delete pending bit
               00   E0   0C23  2554         BBS     #TQE$V_BSY,-
          03 10 A0        0C25  2555                 TQE$L_FR3(R0),7$        ; Branch if busy, will delete in timer routi
             006C   30   0C28  2556         BSBW    50$                     ; Return to pool
                        0C2B  2557 ;
                        0C2B  2558 ;       We must be at ipl 7 or above here
                        0C2B  2559 ;
                        0C2B  2560
                        0C2B  2561 7$:     ASSUME IPL$_TIMER EQ 8
                        0C2B  2562
                        0C2B  2563 ;
                        0C2B  2564 ;       Fix the interlock with the receive iirp so it will be deallocated
                        0C2B  2565 ;       when it completes.  We must say we did so here.  The condition is
                        0C2B  2566 ;       NETIRP = 1 and IRP$L_AST = 0 means that its gone.  If NETIRP = 0
                        0C2B  2567 ;       it has never been allocated and given to netdriver.
                        0C2B  2568 ;
                        0C2B  2569
    50   00C0 C5   D0   0C2B  2570         MOVL    UCB$L_CT_RIIRP(R5),R0   ; Look at address of receive iirp
              06   13   0C30  2571         BEQL    10$                     ; Nope not here
            03 50   E8   0C32  2572         BLBS    R0,10$                  ; Dummy, all done?
            10 A0   D4   0C35  2573         CLRL    IRP$L_AST(R0)           ; Nope so tell receive iirp
    00C0 C5    01   D0   0C38  2574 10$:     MOVL    #1,UCB$L_CT_RIIRP(R5)   ; Clobber address here
                        0C3D  2575 ;
                        0C3D  2576 ; Check if WIIRP is busy writing to net
                        0C3D  2577 ;
    50   00E0 C5   D0   0C3D  2578         MOVL    UCB$L_CT_WIIRP(R5),R0   ; Get address
              10   13   0C42  2579         BEQL    25$                     ; Branch if none
            10 A0   D4   0C44  2580         CLRL    IRP$L_AST(R0)           ; Signal net dead, UCB gone
         00E0 C5   D4   0C47  2581         CLRL    UCB$L_CT_WIIRP(R5)      ; zap pointer
               00   E0   0C4B  2582         BBS     #FLG$V_WIIRP_BSY,-
      03 00DC C5        0C4D  2583                 UCB$W_CT_FLAGS(R5),25$  ; Branch if writing to net
                        0C51  2584 20$:
             0043   30   0C51  2585         BSBW    50$                     ; WIIRP not busy, Return to pool
                        0C54  2586 ;
                        0C54  2587 ;       Now we abort all of the queued irps that we have at this time.
                        0C54  2588 ;
                        0C54  2589 25$:
    53   00B8 D5   0F   0C54  2590         REMQUE  @UCB$L_RTT_IRPFL(R5),R3 ; Obtain an irp from queue
              15   1D   0C59  2591         BVS     30$                     ; No more
      38 A3   2C   3C   0C5B  2592         MOVZWL  #SS$_ABORT, -           ; Complete with abort status
                        0C5F  2593                 IRP$C_IOST1(R3)         ;
```

```
                                                H 16
                3C A3   D4  0C5F  2594          CLRL    IRP$L_IOST2(R3)         ;
                            0C62  2595          POST_IO                         ; post I/O
            50   04     9A  0C62                MOVZBL  #POST_COUNT,R0          ; set position
                 FE85   30  0C65                BSBW    CHECK_POST_IO           ; check for error, set posted
 00000000'GF        16  0C68                    JSB     G^COM$POST              ; post I/O
                 E4     11  0C6E  2596          BRB     25$                     ; and back for more irps
                            0C70  2597
50   00E8 D5        0F  0C70  2598 30$:         REMQUE  @UCB$L_CT_NETQFL(R5),R0 ; Obtain a CTP from queue
                    04  1D  0C75  2599          BVS     40$                     ; No more
                    1E  10  0C77  2600          BSBB    50$                     ; delete
                    F5  11  0C79  2601          BRB     30$                     ; and back for more irps
                            0C7B  2602          ;
                            0C7B  2603          ; Abort STALLED IRP queue
                            0C7B  2604          ;
                            0C7B  2605 40$:
53   00F0 D5        0F  0C7B  2606          REMQUE  @UCB$L_CT_STALLQFL(R5),R3 ; Get stalled IRP
                    1C  1D  0C80  2607          BVS     60$                     ; branch if done
  38 A3             2C  3C  0C82  2608          MOVZWL  #SS$_ABORT, -           ; Complete with abort status
                            0C86  2609                  IRP$L_IOST1(R3)         ;
                3C A3   D4  0C86  2610          CLRL    IRP$L_IOST2(R3)         ;
                            0C89  2611          POST_IO                         ; post I/O
            50   05     9A  0C89                MOVZBL  #POST_COUNT,R0          ; set position
                 FE5E   30  0C8C                BSBW    CHECK_POST_IO           ; check for error, set posted
 00000000'GF        16  0C8F                    JSB     G^COM$POST              ; post I/O
                 E4     11  0C95  2612          BRB     40$                     ; Loop until done
                            0C97  2613
                            0C97  2614 ;
                            0C97  2615 ; Local routine to return buffer to pool
                            0C97  2616 ;
 00000000'GF        16  0C97  2617 50$:         JSB     G^EXE$DEANONPAGED       ; Return to pool
                    05  0C9D  2618          RSB                             ; Exit
                            0C9E  2619 60$:
                    03  E0  0C9E  2620          BBS     #FLG$V_INWRTFDT,-
              0D 00DC C5        0CA0  2621                  UCB$W_CT_FLAGS(R5),80$  ; if FDT in progress, exit
50   00F8 C5        D0  0CA4  2622          MOVL    UCB$L_CT_WRTCTP(R5),R0  ; Fetch multiple write buffer
                    06  13  0CA9  2623          BEQL    80$                     ; branch if none exists
                    EA  10  0CAB  2624          BSBB    50$                     ; Dispose of buffer
                            0CAD  2625          ASSUME  UCB$L_CT_WRTCTP+4 EQ UCB$L_CT_WRTCUR
                00F8 C5     7C  0CAD  2626          CLRQ    UCB$L_CT_WRTCTP(R5)     ; Clear buffer addresses
                            0CB1  2627 ;
                            0CB1  2628 ;        If there are no more channels to this device, then pass it on
                            0CB1  2629 ;        to the acp for disposal.
                            0CB1  2630 ;
                5C A5       B5  0CB1  2631 80$:         TSTW    UCB$W_REFC(R5)          ; Any channels to device?
                    26      12  0CB4  2632          BNEQ    100$                    ; Yes
                            0CB6  2633
                    01      AA  0CB6  2634          BICW    #UCB$M_JOB,-            ; Clear Job Controller notified
                68 A5           0CB8  2635                  UCB$W_DEVSTS(R5)        ;
                    15      E2  0CBA  2636          BBSS    #DEV$V_DMT,-            ; If set, UCB already queued
              1D 38 A5          0CBC  2637                  UCB$L_DEVCHAR(R5),100$  ;
                 53     55      D0  0CBF  2638          MOVL    R5,R3                   ; Set up ucb as the packet
               52   34 A5      D0  0CC2  2639          MOVL    UCB$L_VCB(R5),R2        ; Get address of VCB
               52   10 A2      D0  0CC6  2640          MOVL    VCB$L_AQB(R2),R2        ; Get address of ACP AQB
 00000000'GF        16  0CCA  2641          JSB     G^EXE$INSERTIRP         ; Insert UCB in ACP queue
                    0A  12  0CD0  2642          BNEQ    100$                    ; If neg, not first entry in queue
               51   0C A2      D0  0CD2  2643          MOVL    AQB$L_ACPPID(R2),R1     ; Get ACP process ID
 00000000'GF        16  0CD6  2644          JSB     G^SCH$WAKE              ; Wake the ACP process
```

```
                OCDC  2645
                OCDC  2646 ;***************** DEBUG CODE ***
                OCDC  2647 .IF DF DEBUG_LOG
                OCDC  2648          IFNOTOPT LOGGING,85$                ; branch if not logging
                OCDC  2649          PUSHR   #^M<R6,R7,R8>               ; save
                OCDC  2650          CLRQ    R6                          ; no length or address
                OCDC  2651          MOVL    #3,R8                       ; abort net code
                OCDC  2652          BSBW    TTY$LOG_IO                  ; log to mailbox
                OCDC  2653          POPR    #^M<R6,R7,R8>               ; save
                OCDC  2654 85$:
                OCDC  2655 .ENDC
                OCDC  2656 ;***************** DEBUG CODE ***
                OCDC  2657
                OCDC  2658 100$:
                OCDC  2659          ENBINT                              ; Restore IPL
          05    OCDF  2660          RSB                                 ;
                OCE0  2661
```

```
                        0CE0  2663          .SBTTL CT_WRITE_WRTCTP - Send WRTCTP to NET
                        0CE0  2664
                        0CE0  2665 ;
                        0CE0  2666 ; CT_WRITE_WRTCTP - Send message if it exists
                        0CE0  2667 ;
                        0CE0  2668 ; Inputs:
                        0CE0  2669 ;       R4 - PCB
                        0CE0  2670 ;       R5 - CT UCB
                        0CE0  2671 ;
                        0CE0  2672 ; R0-R1 Scratch
                        0CE0  2673 ;
                        0CE0  2674
                        0CE0  2675 CT_WRITE_WRTCTP:
                        0CE0  2676
                        0CE0  2677          DSBINT  #CT$K_FIPL                  ; synchronize
          013C 8F   BB  0CE6  2678          PUSHR   #^M<R2,R3,R4,R5,R8>         ; Save
          50   01   D0  0CEA  2679          MOVL    #1,R0                       ; Assume success
     52   00F8 C5   D0  0CED  2680          MOVL    UCB$L_CT_WRTCTP(R5),R2      ; Fetch CTP address
               1B   13  0CF2  2681          BEQL    20$                         ; Branch if none
53   00FC C5   52   C3  0CF4  2682          SUBL3   R2,-
                        0CFA  2683                  UCB$L_CT_WRTCUR(R5),R3      ; Calculate size of buffer
          53   02   C0  0CFA  2684          ADDL2   #2,R3                       ; Plus two for foundation type
        14 A2  53   B0  0CFD  2685          MOVW    R3,CTP$W_DATSIZE(R2)        ; Set size
                        0D01  2686 ;
                        0D01  2687 ; Clean up
                        0D01  2688 ;
          00F8 C5   D4  0D01  2689          CLRL    UCB$L_CT_WRTCTP(R5)         ; Clear address
                        0D05  2690          ASSUME  UCB$L_CT_WRTCUR+4 EQ UCB$W_CT_WRTSIZ
                        0D05  2691          ASSUME  UCB$L_CT_WRTCUR+6 EQ UCB$W_CT_WRTCNT
          00FC C5   7C  0D05  2692          CLRQ    UCB$L_CT_WRTCUR(R5)         ; Clear address, count, and offset
          0120 D5   D6  0D09  2693          incl    @ucb$l_cnt_addr(r5)         ; %%% performance hook
                        0D0D  2694 ;
                        0D0D  2695 ; Write to net
                        0D0D  2696 ;
               4B   10  0D0D  2697          BSBB    CT_NET_Q_MSG                ; Send the message
                        0D0F  2698 20$:
     013C 8F    BA  0D0F  2699          POPR    #^M<R2,R3,R4,R5,R8>         ; Restore
                        0D13  2700          ENBINT                              ; Lower IPL
                   05   0D16  2701          RSB
                        0D17  2702
```

```
                                0D17  2704          .SBTTL CT_NETMSGSEND  - Send message to net driver
                                0D17  2705 ;
                                0D17  2706 ; CT_NETMSGSENDX  -  Send message to netdriver and exit qio
                                0D17  2707 ; CT_NET_Q_MSG    -  Send message to netdriver without IRP
                                0D17  2708 ; CT_NET_WRITE    -  Write message to net (through NETDRIVER)
                                0D17  2709 ;
                                0D17  2710 ; inputs:
                                0D17  2711 ;       r2 - address beyond message data (maybe this should be CTP,
                                0D17  2712 ;                               R0 is address of one byte past...***)
                                0D17  2713 ;       r3 - CT irp
                                0D17  2714 ;       r4 - pcb
                                0D17  2715 ;       r5 - CT ucb
                                0D17  2716 ;
                                0D17  2717 ; R0-R2 Destroyed
                                0D17  2718 ;
                                0D17  2719
                                0D17  2720 CT_NETMSGSENDX:          ; Called from FDT level routines
                                0D17  2721
        50    2C A3    D0      0D17  2722          MOVL    IRP$L_SVAPTE(R3),R0     ; The buffer address
              2C A3    D4      0D1B  2723          CLRL    IRP$L_SVAPTE(R3)        ; Clear buffer address
              41 A3    94      0D1E  2724          CLRB    IRP$B_CT_CANCEL(R3)     ; No cancel has been sent yet
  51  52    0C A0    C3      0D21  2725          SUBL3   CTP$L_MSGDAT(R0),R2,R1  ; Make the length of the data
        14 A0    51    B0      0D26  2726          MOVW    R1,CTP$W_DATSIZE(R0)    ; save in the buffer
                 04    A3      0D2A  2727          SUBW3   #<CTP$B_MSGTYPE-CTP$B_PRO_MSGTYPE>,-
        28 A0    51            0D2C  2728                  R1,CTP$W_MSGSIZE(R0)    ; Fill in size of single message
              40 A3    95      0D2F  2729          TSTB    IRP$B_CT_RESPTYPE(R3)   ; See if response from SERVER expected
                 05    13      0D32  2730          BEQL    10$                     ; Branch if not
                 63    0E      0D34  2731          INSQUE  (R3),-
           00BC  D5            0D36  2732                  @UCB$L_RTT_IRPBL(R5)    ; Queue irp
                               0D39  2733 10$:
                 53    DD      0D39  2734          PUSHL   R3                      ; Save IRP
              52 50    D0      0D3B  2735          MOVL    R0,R2                   ; Set CTP address
                               0D3E  2736          DSBINT  #CT$K_FIPL              ; Do this work at driver ipl
                 14    10      0D44  2737          BSBB    CT_NET_Q_MSG            ; Send the message and
                               0D46  2738          ENBINT                          ; Enable interrupts
                 53 8ED0       0D49  2739          POPL    R3                      ; restore IRP
              40 A3    95      0D4C  2740          TSTB    IRP$B_CT_RESPTYPE(R3)   ; See if response from SERVER expected
                 06    13      0D4F  2741          BEQL    20$                     ; Branch if not
     00000000'GF    17      0D51  2742          JMP     G^EXE$QIORETURN         ; Return from the qio
           FBBB    31      0D57  2743 20$:     BRW     FDT_FINISHIOC_OK        ; Finish I/O
                               0D5A  2744
                               0D5A  2745 ;
                               0D5A  2746 ; Inputs:
                               0D5A  2747 ;       R2 - CTP
                               0D5A  2748 ;       R5 - CT UCB
                               0D5A  2749 ;
                               0D5A  2750 ;       IPL = ??? Driver ipl, right? (check all callers)
                               0D5A  2751 ;
                               0D5A  2752 CT_NET_Q_MSG:
                               0D5A  2753
                               0D5A  2754 .if df debug
                               0D5A  2755          movw    ctp$w_datsize(R2),R0    ; ** debug
                               0D5A  2756          addw2   #ctp$b_pro_msgtype,R0   ; ** debug
                               0D5A  2757          cmpw    8(R2),R0                ; ** debug check size (should be equal)
                               0D5A  2758          bgequ   10$                     ; ** debug br if ok
                               0D5A  2759          jsb     G^ini$brk               ; ** debug
                               0D5A  2760 10$:
```

```
                                0D5A  2761 .endc
              37 00C0 C5    E8  0D5A  2762          BLBS    UCB$L_CT_RIIRP(R5),Q_ERR ; If LBS, Net has hungup
                    00      E3  0D5F  2763          BBCS    #FLG$V_WIIRP_BSY,-
              0D 00DC C5        0D61  2764                  UCB$W_CT_FLAGS(R5),20$  ; Branch if not busy, set busy flag
                                0D65  2765 ;
                                0D65  2766 ; Queue message until netdriver returns IIRP
                                0D65  2767 ;
           00EC D5    62    0E  0D65  2768          INSQUE  (R2), -
                                0D6A  2769                  @UCB$L_CT_NETQBL(R5)    ; Queue request on the ucb
                00DE C5    B6  0D6A  2770          INCW    UCB$W_CT_QCTPCNT(R5)    ; Increment count
                50    01    D0  0D6E  2771          MOVL    #1,R0                   ; Set status
                            05  0D71  2772          RSB                             ; return
                                0D72  2773 20$:
                                0D72  2774
                                0D72  2775 CT_NET_WRITE:
                                0D72  2776 ;
                                0D72  2777 ;       R2                      - CTP
                                0D72  2778 ;       R5                      - CT UCB
                                0D72  2779 ;       CTP$W_DATSIZE           - size of message to write
                                0D72  2780 ;
            53    00E0 C5    D0  0D72  2781          MOVL    UCB$L_CT_WIIRP(R5),R3   ; Get address of IIRP
                                0D77  2782          ASSUME  CTP$L_MSGDAT+4 EQ CTP$L_USRBFR
              62    0C A2    7D  0D77  2783          MOVQ    CTP$L_MSGDAT(R2),(R2)   ; Move to first two longwords of structure
              2C A3    52    D0  0D7B  2784          MOVL    R2,IRP$L_SVAPTE(R3)     ; Set address in IIRP
           32 A3    14 A2    B0  0D7F  2785          MOVW    CTP$W_DATSIZE(R2), -    ; Set the size of the buffer
                                0D84  2786                  IRP$W_BCNT(R3)          ;
                                0D84  2787
                                0D84  2788 ;***************** DEBUG CODE ***
                                0D84  2789 .IF DF DEBUG_LOG
                                0D84  2790          IFNOTOPT LOGGING,15$            ; branch if not logging
                                0D84  2791          PUSHR   #^M<R6,R7,R8>           ; save
                                0D84  2792          MOVZWL  IRP$W_BCNT(R3),R6       ; length
                                0D84  2793          MOVL    @IRP$L_SVAPTE(R3),R7    ; address
                                0D84  2794          MOVL    #1,R8                   ; write to net code
                                0D84  2795          BSBW    TTY$LOG_IO              ; log to mailbox
                                0D84  2796          POPR    #^M<R6,R7,R8>           ; save
                                0D84  2797 15$:
                                0D84  2798 .ENDC
                                0D84  2799 ;***************** DEBUG CODE ***
                                0D84  2800
                      38    BB  0D84  2801          PUSHR   #^M<R3,R4,R5>           ; Save the magic three
               55    1C A3    D0  0D86  2802          MOVL    IRP$L_UCB(R3),R5        ; The netucb from this packet
           00000000'GF    16  0D8A  2803          JSB     G^EXE$ALTQUEPKT         ; Queue iirp to netdriver
                      38    BA  0D90  2804          POPR    #^M<R3,R4,R5>           ; restore magic three
                  50    01    D0  0D92  2805          MOVL    #1,R0                   ; return success
                            05  0D95  2806          RSB
                                0D96  2807 Q_ERR:
                      52    DD  0D96  2808          PUSHL   R2                      ; Save CTP
                    FE73    30  0D98  2809          BSBW    CT_ABORTIRPS            ; Clean up queues (again)
                      50 8ED0   0D9B  2810          POPL    R0                      ; restore CTP
           00000000'GF    16  0D9E  2811          JSB     G^EXE$DEANONPAGED       ; Get rid of the buffer
                            05  0DA4  2812          RSB
                                0DA5  2813
```

```
                        0DA5  2815          .SBTTL  CT_NETWRTDONE  - Post routine for net write
                        0DA5  2816 ;
                        0DA5  2817 ; CT_NETWRTDONE
                        0DA5  2818 ;
                        0DA5  2819 ;       Called here via JSB from IOPOST because PID field in IIRP is
                        0DA5  2820 ;       negative (i.e. the address of this routine).
                        0DA5  2821 ;
                        0DA5  2822 ;       *** This routine has been coded so that there is no dependance
                        0DA5  2823 ;       *** on data in the CTP. This is because NETDRIVER may soon learn
                        0DA5  2824 ;       *** to use the CTP as it's own buffer and the IIRP will return
                        0DA5  2825 ;       *** here with SVAPTE as zero...
                        0DA5  2826 ;
                        0DA5  2827 ;       Enter here to post writes to net.
                        0DA5  2828 ;       Deallocate the message buffer if any.
                        0DA5  2829 ;
                        0DA5  2830 ;       r5 -> write iirp
                        0DA5  2831 ;       ipl = ipl$_iopost (is CT$K_FIPL ok for mucking with STALL Q???)
                        0DA5  2832 ;
                        0DA5  2833 ;       R0-R5 are scratch, return to IOPOST.
                        0DA5  2834 ;
                        0DA5  2835 ;
                        0DA5  2836
                        0DA5  2837 CT_NETWRTDONE:
                        0DA5  2838
                        0DA5  2839          ;
                        0DA5  2840          ; Status of the write is not checked. This is
                        0DA5  2841          ; because if the link fails,
                        0DA5  2842          ; it will be reported via the read queued at CT_NETREADDONE.
                        0DA5  2843          ;
                        0DA5  2844          DSBINT  #CT$K_FIPL                  ; Set ipl
     50    2C A5    D0  0DAB  2845          MOVL    IRP$L_SVAPTE(R5),R0         ; Get buffer address if it exists
           09       13  0DAF  2846          BEQL    10$                         ; Branch if it doesn't
        2C A5       D4  0DB1  2847          CLRL    IRP$L_SVAPTE(R5)            ; clear pointer
00000000'GF         16  0DB4  2848          JSB     G^COM$DRVDEALMEM            ; Deallocate buffer
                        0DBA  2849 10$:
     52    55       D0  0DBA  2850          MOVL    R5,R2                       ; Move IIRP address
     55    10 A2    D0  0DBD  2851          MOVL    IRP$L_AST(R2),R5            ; Get UCB address
           39       13  0DC1  2852          BEQL    150$                        ; If zero, net hungup during write
                        0DC3  2853          ;
                        0DC3  2854          ; See if there is another message to write
                        0DC3  2855          ;
                        0DC3  2856 100$:
     52    00E8 D5    0F  0DC3  2857          REMQUE  @UCB$L_CT_NETQFL(R5),R2 ; Another packet?
              10       1D  0DC8  2858          BVS     110$                        ; Branch if no
        00DE C5       B7  0DCA  2859          DECW    UCB$W_CT_QCTPCNT(R5)        ; subtract one
           FFA1       30  0DCE  2860          BSBW    CT_NET_WRITE                ; Write to net
              01       B1  0DD1  2861          CMPW    #CT$K_CTPLOLIM,-
        00DE C5           0DD3  2862                  UCB$W_CT_QCTPCNT(R5)        ; Queue getting low
              20       19  0DD6  2863          BLSS    140$                        ; Exit
              09       11  0DD8  2864          BRB     120$                        ; complete stalled IRP
                        0DDA  2865          ;
                        0DDA  2866          ; Writing to net has emptied CTP queue, check if there are stalled IRPS...
                        0DDA  2867          ;
                        0DDA  2868 110$:
        00DE C5       B4  0DDA  2869          CLRW    UCB$W_CT_QCTPCNT(R5)        ; Clear (should be redundant)
              01       8A  0DDE  2870          BICB    #FLG$M_WIIRP_BSY,-
        00DC C5           0DE0  2871                  UCB$W_CT_FLAGS(R5)          ; Clear busy flag
```

```
                                                B 1
                    0DE3  2872 120$:
53   00F0 D5   0F   0DE3  2873          REMQUE  @UCB$L_CT_STALLQFL(R5),R3 ; Get stalled IRP
          0E   1D   0DE8  2874          BVS     140$                       ; branch if done
               0DEA  2875          POST_IO                            ; post I/O
     50   06   9A   0DEA                MOVZBL  #POST_COUNT,R0             ; set position
        FCFD   30   0DED                BSBW    CHECK_POST_IO              ; check for error, set posted
00000000'GF   16   0DF0                JSB     G^COM$POST                 ; post I/O
          EB   11   0DF6  2876          BRB     120$                       ; Loop until done
               0DF8  2877 140$:
               0DF8  2878          ENBINT                             ; Back to ipl$_iopost
               05   0DFB  2879          RSB
               0DFC  2880
               0DFC  2881 ;
               0DFC  2882 ; Net has hung up while doing write, return WIIRP to pool, (UCB is gone?)
               0DFC  2883 ;
               0DFC  2884
               0DFC  2885 150$:
     50   52   D0   0DFC  2886          MOVL    R2,R0                      ; Set address of WIIRP
00000000'GF   16   0DFF  2887          JSB     G^EXE$DEANONPAGED          ; Return to pool
          F1   11   0E05  2888          BRB     140$                       ; exit
               0E07  2889
```

```
                     0E07  2891          .SBTTL  CT_UNSOLIC Unsolicited interrupt handler
                     0E07  2892 ;++
                     0E07  2893 ; CT_UNSOLIC, Unsolicted interrupt handler
                     0E07  2894 ;
                     0E07  2895 ; Functional description:
                     0E07  2896 ;
                     0E07  2897 ;       This routine handles unsolicted attention messages from the remote
                     0E07  2898 ;       ACP (REMACP).
                     0E07  2899 ;
                     0E07  2900 ; Inputs:
                     0E07  2901 ;
                     0E07  2902 ;       R3 = address of attention message (type = DYN$_BUFIO)
                     0E07  2903 ;            first longword in structure points to data.
                     0E07  2904 ;       R5 = address of UCB
                     0E07  2905 ;
                     0E07  2906 ;       IPL = 0
                     0E07  2907 ;
                     0E07  2908 ; Outputs:
                     0E07  2909 ;
                     0E07  2910 ;       Buffer pointed to by R3 is returned to pool.
                     0E07  2911 ;
                     0E07  2912 ;--
                     0E07  2913 CT_UNSOLIC:                             ; Unsolicted interrupt handler
                     0E07  2914
                     0E07  2915 ; *** some checking of message type should take place here, right???
                     0E07  2916
          53    DD   0E07  2917          PUSHL   R3                      ; Save buffer address
                     0E09  2918          DSBINT  #CT$K_FIPL              ; Synch
        02F7    30   0E0F  2919          BSBW    STARTNETRCV             ; Start up net read
                     0E12  2920          ENBINT                          ; Enable interrupts
          50  8ED0   0E15  2921          POPL    R0                      ; Restore buffer address
00000000'GF    16   0E18  2922          JSB     G^EXE$DEANONPAGED       ; Return to pool
                05   0E1E  2923          RSB                             ; return to REMACP
                     0E1F  2924
```

```
                                                D  1
                     0E1F  2926          .SBTTL  CT_NETREADDONE  - Post routine for net receive
                     0E1F  2927 ;
                     0E1F  2928 ; CT_NETREADDONE  Post net receive
                     0E1F  2929 ;
                     0E1F  2930 ;       This is the post routine for receives from the netdriver.
                     0E1F  2931 ;       If the type is not recognised or we can't find the irp,
                     0E1F  2932 ;       we hangup the terminal.
                     0E1F  2933 ;
                     0E1F  2934 ;       We are going to run this code at CT driver ipl.
                     0E1F  2935 ;
                     0E1F  2936 ; inputs:
                     0E1F  2937 ;       r5 -> net iirp
                     0E1F  2938 ;       ipl = ipl$_iopost
                     0E1F  2939 ;
                     0E1F  2940 ;       R0-R5 are scratch, return to IOPOST.
                     0E1F  2941 ;
                     0E1F  2942
                     0E1F  2943 CT_NETREADDONE:
                     0E1F  2944
                     0E1F  2945          DSBINT  #CT$K_FIPL                  ; Do this work at driver ipl
          55    DD   0E25  2946          PUSHL   R5                          ; Save net read IIRP address
     53   55    D0   0E27  2947          MOVL    R5,R3                       ; The iirp address is here
 55     10 A3    D0   0E2A  2948          MOVL    IRP$L_AST(R3),R5            ; The CT ucb?
          22    13   0E2E  2949          BEQL    20$                         ; Its gone, we are hung up
   1B 38 A3    E9   0E30  2950          BLBC    IRP$L_IOST1(R3),15$         ; Error? if so then hang up
52     2C A3    D0   0E34  2951          MOVL    IRP$L_SVAPTE(R3),R2         ; The buffer address
          26    C3   0E38  2952          SUBL3   #CTP$B_PRO_MSGTYPE,-
     51   62          0E3A  2953                  (R2),R1                     ; Point to message
                     0E3C  2954          ;
                     0E3C  2955          ; Consistency check - make sure we really only get one
                     0E3C  2956          ; message here
                     0E3C  2957          ;
50     28 A1    3C   0E3C  2958          MOVZWL  CTP$W_MSGSIZE(R1),R0        ; length of message
     50   04    C0   0E40  2959          ADDL    #4,R0                       ; Plus 4 for found overhead & msg size
32 A3    50    B1   0E43  2960          CMPW    R0,IRP$W_BCNT(R3)           ; compare to DECnet message length
          0C    13   0E47  2961          BEQL    5$                          ; Branch if OK
                     0E49  2962          MINOR_ERROR                         ; increment error count
          06    11   0E4D  2963          BRB     5$                          ; continue
                     0E4F  2964
        022D    31   0E4F  2965 15$:     BRW     FOUND_ERROR                 ; error on net, must hangup
        022D    31   0E52  2966 20$:     BRW     FOUND_ERROR_2               ; error, hangup already done
                     0E55  2967
                     0E55  2968 5$:
                     0E55  2969 ;**************** DEBUG CODE ***
                     0E55  2970 .IF DF DEBUG_LOG
                     0E55  2971          IFNOTOPT LOGGING,7$                 ; branch if not logging
                     0E55  2972          PUSHR   #^M<R6,R7,R8>               ; save
                     0E55  2973          MOVZWL  IRP$W_BCNT(R3),R6           ; length
                     0E55  2974          MOVL    (R2),R7                     ; address
                     0E55  2975          MOVL    #2,R8                       ; write to net code
                     0E55  2976          BSBW    TTY$LOG_IO                  ; log to mailbox
                     0E55  2977          POPR    #^M<R6,R7,R8>               ; save
                     0E55  2978 7$:
                     0E55  2979 .ENDC
                     0E55  2980 ;**************** DEBUG CODE ***
                     0E55  2981
                     0E55  2982          CASE    CTP$B_PRO_MSGTYPE(R1),- ; case on message type
```

```
                0E55  2983                  <10$,-                          ; 0 is illegal
                0E55  2984                   FOUND_BIND,-
                0E55  2985                   FOUND_UNBIND,-
                0E55  2986                   FOUND_REBIND,-
                0E55  2987                   FOUND_ACCEPT,-
                0E55  2988                   FOUND_ENTR_MODE,-
                0E55  2989                   FOUND_EXIT_MODE,-
                0E55  2990                   FOUND_CONF_MODE,-
                0E55  2991                   FOUND_NO_MODE,-
                0E55  2992                   FOUND_CTERM,-
                0E55  2993                   FOUND_MODE>,-                   ; type 10 not used yet
                0E55  2994                  TYPE = B
                0E70  2995
                0E70  2996 10$:     MINOR_ERROR                             ; increment error count
01E0     31     0E74  2997          BRW     FOUND_EXIT                      ; requeue read
                0E77  2998
```

```
                     0E77  3000 .SBTTL  FOUNDATION - Miscellaneous foundation message
                     0E77  3001
                     0E77  3002 FOUND_BIND:
                     0E77  3003 ;
                     0E77  3004 ;       Start the first receive iirp to the netdriver.  We make an iirp
                     0E77  3005 ;       and queue it to the netdriver with a read function in it.
                     0E77  3006 ;       Also, the write IIRP is allocated and the address is saved in the
                     0E77  3007 ;       UCB for future reference.
                     0E77  3008 ;
                     0E77  3009 ; inputs:
                     0E77  3010 ;       R3 - message buffer (IRP)
                     0E77  3011 ;       R5 - CT ucb
                     0E77  3012 ;
                     0E77  3013
028F   30            0E77  3014         BSBW    STARTNETRCV             ; will we ever get here?
01DA   31            0E7A  3015         BRW     FOUND_EXIT              ; exit
                     0E7D  3016
                     0E7D  3017 FOUND_UNBIND:
                     0E7D  3018 ;
                     0E7D  3019 ; Deliver hangup notification ??? UNBIND ???
                     0E7D  3020 ;
                     0E7D  3021 HANGUP:                                 ; Dataset hangup
FD56   30            0E7D  3022         BSBW    CT_HANGUP               ; Do the hangup stuff
01D4   31            0E80  3023         BRW     FOUND_EXIT
                     0E83  3024
                     0E83  3025 FOUND_REBIND:                           ; Ignore all these messages for now
                     0E83  3026 FOUND_ACCEPT:
                     0E83  3027 FOUND_ENTR_MODE:
                     0E83  3028 FOUND_EXIT_MODE:
                     0E83  3029 FOUND_CONF_MODE:
                     0E83  3030 FOUND_NO_MODE:
                     0E83  3031 FOUND_MODE:
                     0E83  3032
01D1   31            0E83  3033         BRW     FOUND_EXIT
                     0E86  3034
```

```
                                                G 1
0E86  3036 .SBTTL  FOUND_CTERM - Foundation CTERM message
0E86  3037
0E86  3038 FOUND_CTERM:
0E86  3039
0E86  3040 ;
0E86  3041 ; Legal message types are:
0E86  3042 ;
0E86  3043 ; 1  - init
0E86  3044 ; 3  - read data        +
0E86  3045 ; 4  - out of band
0E86  3046 ; 8  - write complete   +
0E86  3047 ; 9  - discard state
0E86  3048 ; 11 - characteristics  +
0E86  3049 ; 13 - input count      +
0E86  3050 ; 14 - input state
0E86  3051 ;
0E86  3052 ; Those labeled above with "+" are expected data, i.e. they have an IRP
0E86  3053 ; waiting in the CT ucb queue. The others are unsolicited data.
0E86  3054 ;
0E86  3055
0E86  3056         CASE    CTP$B_MSGTYPE(R1),-
0E86  3057                 <10$,-                  ; 0
0E86  3058                 CTERM_INIT,10$,-        ; 1,2
0E86  3059                 20$,-                   ; 3
0E86  3060                 CTERM_OUTBAND,-         ; 4
0E86  3061                 10$,10$,10$,-           ; 5,6,7
0E86  3062                 20$,-                   ; 8
0E86  3063                 CTERM_DISCARD,-         ; 9
0E86  3064                 10$,20$,-               ; 10,11
0E86  3065                 10$,20$,-               ; 12,13
0E86  3066                 CTERM_IN_STATE,-        ; 14
0E86  3067                 VMS_QIO,-               ; 15 (VMS specific)
0E86  3068                 VMS_BRDCST-             ; 16 (VMS specific)
0E86  3069                 >-
0E86  3070                 TYPE = B
0EAD  3071 ;
0EAD  3072 ; Unknown
0EAD  3073 ;
0EAD  3074 10$:
      01A7  31  0EAD  3075         BRW     FOUND_EXIT
0EB0  3076
0EB0  3077 ;
0EB0  3078 ; IRP queue is assumed to be in order for reads, characteristics,
0EB0  3079 ; and input counts. All these packets have the FUNC bit set in the
0EB0  3080 ; queued IRP. The write packets can complete asynchronously to all
0EB0  3081 ; these other types, and they do not have the FUNC bit set. Search
0EB0  3082 ; for the associated IRP:
0EB0  3083 ;
0EB0  3084
0EB0  3085 20$:
54    00B8 C5   7E  0EB0  3086         MOVAQ   UCB$L_RTT_IRPFL(R5),R4  ; Look through the irps for ours
      50   54   D0  0EB5  3087         MOVL    R4,R0                   ; head of queue here
0EB8  3088
      54   64   D0  0EB8  3089 30$:    MOVL    (R4),R4                 ; Link through chain
      50   54   D1  0EBB  3090         CMPL    R4,R0                   ; end of irps?
           19   13  0EBE  3091         BEQL    40$                     ; Yes, could not find it, ignore it
0EC0  3092         ; NB: this error used to abort link, now it only gets ignored.
```

```
                                                H  1
                        0EC0  3093          ; this is because of a change to CANCEL that could potentially
                        0EC0  3094          ; zpa the IRP in the waiting list.
40 A4   2A A1   91      0EC0  3095          CMPB    CTP$B_MSGTYPE(R1), -
                        0EC5  3096                  IRP$B_CT_RESPTYPE(R4)   ; Does it match?
            F1  12      0EC5  3097          BNEQ    30$                     ; Branch if no
                        0EC7  3098 ;
                        0EC7  3099 ; A match has been found
                        0EC7  3100 ;
        2C A3   D4      0EC7  3101          CLRL    IRP$L_SVAPTE(R3)        ; Buffer not in net iirp now
        32 A3   B0      0ECA  3102          MOVW    IRP$W_BCNT(R3),-
        3C A4           0ECD  3103                  IRP$L_IOST2(R4)         ; Set size of buffer read from net
     53   64   0F      0ECF  3104          REMQUE  (R4),R3                 ; Remove the CT irp from queue
  2C A3   52   D0      0ED2  3105          MOVL    R2,IRP$L_SVAPTE(R3)     ; stick buffer there
        FAB6   30      0ED6  3106          BSBW    CT_INTERRUPT            ; and call interrupt routine
                        0ED9  3107 40$:
        017B   31      0ED9  3108          BRW     FOUND_EXIT
                        0EDC  3109
```

```
                                                     I  1
                     0EDC  3111 CTERM_INIT:
                     0EDC  3112          ;
                     0EDC  3113          ; Parse INIT message, set up parameters
                     0EDC  3114          ;
                     0EDC  3115          ASSUME  CTP$B_IN_VERSION+1 EQ CTP$B_IN_ECO
                     0EDC  3116          ASSUME  UCB$B_CT_VERSION+1 EQ UCB$B_CT_ECO
          2C A1  B0  0EDC  3117          MOVW    CTP$B_IN_VERSION(R1),-
        010C C5      0EDF  3118                  UCB$B_CT_VERSION(R5)    ; Save version and ECO
                     0EE2  3119
        07C0 8F  BB  0EE2  3120          PUSHR   #^M<R6,R7,R8,R9,R10>    ; Save some work registers
    56    37 A1  9E  0EE6  3121          MOVAB   CTP$B_IN_PARMTYPE(R1),R6 ; address of first parameter
    57    2A A1  9E  0EEA  3122          MOVAB   CTP$B_MSGTYPE(R1),R7    ; address of start of message
    5A    28 A1  3C  0EEE  3123          MOVZWL  CTP$W_MSGSIZE(R1),R10   ; length of message
        5A   57  C0  0EF2  3124          ADDL    R7,R10                  ; calculate end of message
                     0EF5  3125
        56   5A  D1  0EF5  3126 10$:     CMPL    R10,R6                  ; Check against end of message
             5F  15  0EF8  3127          BLEQ    90$                     ; Exit loop if done
    57    01 A6  9A  0EFA  3128          MOVZBL  1(R6),R7                ; Size of parameter
        59   02  D0  0EFE  3129          MOVL    #2,R9                   ; Default maximum storage area size
                     0F01  3130          CASE    (R6),-
                     0F01  3131                  <20$,30$,40$,50$>,-     ; case on selector
                     0F01  3132                  TYPE=B,LIMIT=#1         ; Limit is really base?
                     0F0D  3133          MINOR_ERROR                     ; increment error count
             46  11  0F11  3134          BRB     90$                     ; continue (no attempt to continue parsing)
                     0F13  3135
58  0104 C5  9E  0F13  3136 20$:     MOVAB   UCB$W_CT_MAXMSG(R5),R8  ; Max message that can written to net
             18  11  0F18  3137          BRB     70$
58  0106 C5  9E  0F1A  3138 30$:     MOVAB   UCB$W_CT_MAXREAD(R5),R8 ; Max read buffer server can handle
             11  11  0F1F  3139          BRB     70$
58  0108 C5  9E  0F21  3140 40$:     MOVAB   UCB$L_CT_LEGALMSG(R5),R8; Legal messages
        59   04  D0  0F26  3141          MOVL    #4,R9                   ; Max size to store
             07  11  0F29  3142          BRB     70$
                     0F2B  3143          ;
                     0F2B  3144          ; parameter 4 is VMS-specific
                     0F2B  3145          ;
                     0F2B  3146          ASSUME  UCB$B_DEVCLASS+1  EQ UCB$B_DEVTYPE      ; assume 12 contiguous bytes
                     0F2B  3147          ASSUME  UCB$B_DEVTYPE+1   EQ UCB$W_DEVBUFSIZ
                     0F2B  3148          ASSUME  UCB$W_DEVBUFSIZ+2 EQ UCB$L_DEVDEPEND
                     0F2B  3149          ASSUME  UCB$L_DEVDEPEND+4 EQ UCB$L_DEVDEPND2
  58    40 A5  9E  0F2B  3150 50$:     MOVAB   UCB$B_DEVCLASS(R5),R8   ; address of characteristics, etc.
    59    0C  D0  0F2F  3151          MOVL    #12,R9                  ; length
                     0F32  3152
  56    02 A6  9E  0F32  3153 70$:     MOVAB   2(R6),R6                ; bias
    59    57  D1  0F36  3154          CMPL    R7,R9                   ; check sizes
             10  1B  0F39  3155          BLEQU   75$                     ; branch if it fits
                     0F3B  3156          MINOR_ERROR                     ; increment error count
7E  57    59  C3  0F3F  3157          SUBL3   R9,R7,-(SP)             ; calculate remainder
    57    59  D0  0F43  3158          MOVL    R9,R7                   ; set size
             59 8ED0  0F46  3159          POPL    R9                      ; remainder
             02  11  0F49  3160          BRB     80$                     ;
             59  D4  0F4B  3161 75$:     CLRL    R9                      ; Set no remainder
                     0F4D  3162
    88   86  90  0F4D  3163 80$:     MOVB    (R6)+,(R8)+             ; set up
      FA 57  F5  0F50  3164          SOBGTR  R7,80$                  ; Loop through
  56  6649  9E  0F53  3165          MOVAB   (R6)[R9],R6             ; add remainder
             9C  11  0F57  3166          BRB     10$                     ; Loop
                     0F59  3167 90$:
```

```
                                                J 1
                                0F59  3168          ;
                                0F59  3169          ; Should have LEGALMSG filled in now, check for base cterm
                                0F59  3170          ;
                     0F   00    ED  0F59  3171          CMPZV   #0,#15,-
                     0108 C5        0F5C  3172                  UCB$L_CT_LEGALMSG(R5),-
                00007FFE 8F         0F5F  3173                  #^B01T111T1111111110    ; required messages
                       04   13      0F64  3174          BEQL    95$                     ; branch if ok
                                    0F66  3175          MINOR_ERROR                     ; Minor protocol error
                     07C0 8F    BA  0F6A  3176 95$      POPR    #^M<R6,R7,R8,R9,R10>    ; Restore
                                    0F6E  3177          ;
                                    0F6E  3178          ; Pick up host characteristic SYSPASWORD
                                    0F6E  3179          ;
          48 A5   00080000 8F   CA  0F6E  3180          BICL    #TT2$M_SYSPWD,UCB$L_DEVDEPND2(R5)       ; Clear bit
          08 00000000'GF   13   E1  0F76  3181          BBC     #TT2$V_SYSPWD,G^TTY$GL_DEFCHAR2,97$     ; skip if clear
          48 A5   00080000 8F   C8  0F7E  3182          BISL    #TT2$M_SYSPWD,UCB$L_DEVDEPND2(R5)       ; Set bit
                                    0F86  3183 97$:
                                    0F86  3184          ;
                                    0F86  3185          ; Notify JOB CONTROLER if logins are enabled. This is a difference
                                    0F86  3186          ; from the old vax protocol. A unsolicted data message was generated
                                    0F86  3187          ; in RTPAD to fake a login attempt. TSA defines this differently,
                                    0F86  3188          ; so the login startup is done here. It is a protocol error to
                                    0F86  3189          ; receive more than one INIT message, but that isn't checked here.
                                    0F86  3190          ;
             53 00000000'GF   D0    0F86  3191          MOVL    G^TTY$GL_JOBCTLMB,R3    ; Get address of Job Controller mailbox
                       68 A3  B5    0F8D  3192          TSTW    UCB$W_DEVSTS(R3)        ; Test high bit *** hack, needs symbol
                          03  14    0F90  3193          BGTR    100$                    ; If clear, continue
                        00EA  31    0F92  3194          BRW     FOUND_ERROR             ; hangup and exit, logins are disabled
                                    0F95  3195 100$:
           10 68 A5     00    E0  0F95  3196          BBS     #UCB$V_JOB,UCB$W_DEVSTS(R5),110$ ; Branch if notified already
                 54     01    9A  0F9A  3197          MOVZBL  #MSG$_TRMUNSOLIC,R4     ; Set mailbox message type
              00000000'GF   16    0F9D  3198          JSB     G^EXE$SNDEVMSG          ; Deliver notification to mailbox
                    04 50   E9    0FA3  3199          BLBC    R0,110$                 ; If lbc failure
                 68 A5      01    A8  0FA6  3200          BISW    #UCB$M_JOB,UCB$W_DEVSTS(R5) ; Set Job Controller notified
                                    0FAA  3201 110$:
                        00AA  31    0FAA  3202          BRW     FOUND_EXIT              ; exit
```

```
                     OFAD  3204          ;
                     OFAD  3205          ; Deliver unsolicited data notification
                     OFAD  3206          ;
                     OFAD  3207 CTERM_IN_STATE:                          ; Unsolicited data
                     OFAD  3208
          00   E1   OFAD  3209          BBC     #CTP$V_IS_NONZERO,-
    14 2B A1         OFAF  3210                  CTP$B_IS_FLAGS(R1),10$  ; Branch if not a zero to non-zero change
       5C A5   B5    OFB2  3211          TSTW    UCB$W_REFC(R5)          ; Any references to device?
          0F   13    OFB5  3212          BEQL    10$                     ; If eql no - notify Job Controller
 53   60 A5   D0    OFB7  3213          MOVL    UCB$L_AMB(R5),R3        ; Get address of associated mailbox
          09   13    OFBB  3214          BEQL    10$                     ; If eql none - forget it
    54    01   9A    OFBD  3215          MOVZBL  #MSG$_TRMUNSOLIC,R4     ; Set mailbox message type
00000000'GF   16    OFC0  3216          JSB     G^EXE$SNDEVMSG          ; Deliver notification to mailbox
                     OFC6  3217 10$:
        008E   31    OFC6  3218          BRW     FOUND_EXIT
```

```
                     OFC9  3220 ;
                     OFC9  3221 ; Change output discard state, user has typed ^O.
                     OFC9  3222 ;
                     OFC9  3223 CTERM_DISCARD:
                     OFC9  3224
      02     A8      OFC9  3225          BISW    #FLG$M_CTRLO,-
  00DC C5            OFCB  3226                  UCB$W_CT_FLAGS(R5)      ; Assume Set ^O
      00     E0      OFCE  3227          BBS     #CTP$V_DS_DISCARD,-
0A 2B A1             OFD0  3228                  CTP$B_DS_FLAGS(R1),10$  ; Brarch if output being suppressed
      02     AA      OFD3  3229          BICW    #FLG$M_CTRLO,-
  00DC C5            OFD5  3230                  UCB$W_CT_FLAGS(R5)      ; Clear ^O
      04     A8      OFD8  3231          BISW    #FLG$M_CANCTRLO,-
  00DC C5            OFDA  3232                  UCB$W_CT_FLAGS(R5)      ; Set cancel ^O pending
                     OFDD  3233 10$:
     0077    31      OFDD  3234          BRW     FOUND_EXIT              ; exit
```

```
                                M 1
                     0FE0  3236 CTERM_OUTBAND:
                     0FE0  3237
      53    2C A1    9A 0FE0  3238          MOVZBL  CTP$B_OB_CHAR(R1),R3     ; Deliver the asts (char)
18 0098 C5    53    E1 0FE4  3239          BBC     R3,UCB$L_TL_OUTBAND(R5),10$ ; branch if no OOB in mask
      54    0098 C5    9E 0FEA  3240          MOVAB   UCB$L_TL_BANDQUE(R5),R4 ; List address
            0042 8F    BB 0FEF  3241          PUSHR   #^M<R1,R6>              ; save CTP (and R6?)
      56    00A4 C5    D0 0FF3  3242          MOVL    UCB$L_TL_CTLPID(R5),R6  ; Set controlling PID
       00000000'GF     16 0FF8  3243          JSB     G^COM$DELCTRLASTP       ; Deliver the asts
            0042 8F    BA 0FFE  3244          POPR    #^M<R1,R6>              ; restore CTP, R6
                     1002  3245 10$:
                     1002  3246 ;
                     1002  3247 ; Check for ^C or ^Y
                     1002  3248 ;
      53    2C A1    9A 1002  3249          MOVZBL  CTP$B_OB_CHAR(R1),R3     ; Deliver the asts (char)
      54    0094 C5    DE 1006  3250          MOVAL   UCB$L_TL_CTRLC(R5),R4   ; Get address of CTRL/C AST list
            03    53   91 100B  3251          CMPB    R3,#TTY$C_CTRLC         ; ^C ?
                  13   12 100E  3252          BNEQ    20$                     ; Branch if not ^C
            0080 8F    A8 1010  3253          BISW    #FLG$M_CTRLC,-
            00DC C5          1014  3254                  UCB$W_CT_FLAGS(R5)      ; Set control C delivered
                  05   E0 1017  3255          BBS     #FLG$V_VAXTOVAX,-
         17 00DC C5          1019  3256                  UCB$W_CT_FLAGS(R5),30$  ; If VAX to VAX, skip next check
                     101D  3257          ;
                     101D  3258          ; connected to TOPS or RSX, must do implicit ^C to ^Y mapping here.
                     101D  3259          ; * the following check is probably wrong when a process
                     101D  3260          ; * has spawned and has a ^C enabled...
                     101D  3261          ;
                  64   D5 101D  3262          TSTL    (R4)                    ; is there a ^C to deliver?
                  07   13 101F  3263          BEQL    25$                     ; turn ^C into ^Y
                  11   11 1021  3264          BRB     30$                     ; deliver ^C
                     1023  3265
            19    53   91 1023  3266 20$:     CMPB    R3,#TTY$C_CTRLY         ; ^Y ?
                  21   12 1026  3267          BNEQ    40$                     ; Branch if not ^Y
            0080 8F    AA 1028  3268 25$:     BICW    #FLG$M_CTRLC,-
            00DC C5          102C  3269                  UCB$W_CT_FLAGS(R5)      ; Set NOT control C delivered
      54    0090 C5    DE 102F  3270          MOVAL   UCB$L_TL_CTRLY(R5),R4   ; Get address of CTRL/Y AST list
                     1034  3271 30$:
                  56   DD 1034  3272          PUSHL   R6                      ; save (Necessary?)
      56    00A4 C5    D0 1036  3273          MOVL    UCB$L_TL_CTLPID(R5),R6  ; Set controlling PID
       00000000'GF     16 103B  3274          JSB     G^COM$DELATTNASTP       ; Deliver the AST's
                  56 8ED0 1041  3275          POPL    R6                      ; restore
                     1044  3276 ;
                     1044  3277 ; If OUT-OF-BAND DISCARD IS TRUE, SERVER IS DISCARD OUTPUT RIGHT NOW. ***
                     1044  3278 ;
                  04   A8 1044  3279          BISW    #FLG$M_CANCTRLO,-
            00DC C5          1046  3280                  UCB$W_CT_FLAGS(R5)      ; Set cancel ^O pending
                     1049  3281 40$:
               000B   31 1049  3282          BRW     FOUND_EXIT
                     104C  3283
```

```
                        104C  3285 VMS_QIO:
          EFB1'   30    104C  3286         BSBW    CT_VMSQIO_MSG           ; VMS qio message
        2D 50     F9    104F  3287         BLBC    R0,FOUND_ERROR          ; hangup if error
           03     11    1052  3288         BRB     FOUND_EXIT              ; otherwise, fall through to found_exit
                        1054  3289
                        1054  3290 VMS_BRDCST:
                        1054  3291
          EFA9'   30    1054  3292         BSBW    CT_VMS_BRDCST           ; VMS upline broadcast for mailbox
                        1057  3293                                         ; fall through to found_exit
                        1057  3294 FOUND_EXIT:
                        1057  3295 ;
                        1057  3296 ;       8(SP)   RTNADR
                        1057  3297 ;       4(SP)   SAVED IPL (iopost)
                        1057  3298 ;       0(SP)   R5 (iirp address)
                        1057  3299 ;
           53 8ED0      1057  3300         POPL    R3                      ; Obtain the net iirp address
     55    1C A3   D0    105A  3301         MOVL    IRP$L_UCB(R3),R5        ; Set the net ucb address up
     50    2C A3   D0    105E  3302         MOVL    IRP$L_SVAPTE(R3),R0     ; dump the buffer
           09      13    1062  3303         BEQL    50$                     ; if there is one to dump
00000000'GF   16    1064  3304         JSB     G^COM$DRVDEALMEM        ; Deallocate buffer
        2C A3   D4    106A  3305         CLRL    IRP$L_SVAPTE(R3)        ; forget it
00000000'GF   80    106D  3306 50$:     MOVW    G^IOC$GW_MAXBUF,-       ; setup for another read from net
        32 A3         1073  3307                 IRP$W_BCNT(R3)          ; with requested buffer size
00000000'GF   16    1075  3308         JSB     G^EXE$ALTQUEPKT         ; queue to net driver
                        107B  3309
                        107B  3310 FOUND_EXIT_2:
                        107B  3311         ENBINT                          ; Restore the ipl
              05    107E  3312         RSB
                        107F  3313 ;
                        107F  3314 ;       If we had on io error in the packet, then hangup the terminal
                        107F  3315 ;       deallocate the packet and any buffer and exit.
                        107F  3316 ;       If there is no CT ucb left anymore, just deallocate the packet
                        107F  3317 ;       and buffer and get out.
                        107F  3318 ;
                        107F  3319
                        107F  3320 FOUND_ERROR:
                        107F  3321
            FB54    30  107F  3322         BSBW    CT_HANGUP               ; Bad error - hangup the terminal
                        1082  3323
                        1082  3324 FOUND_ERROR_2:
                        1082  3325
              55 8ED0   1082  3326         POPL    R5                      ; Restore Read IIRP
                        1085  3327 ;
                        1085  3328 ; Deallocate IIRP and buffer
                        1085  3329 ;
     50    2C A5   D0    1085  3330         MOVL    IRP$L_SVAPTE(R5),R0     ; Buffer on this iirp?
           06      13    1089  3331         BEQL    10$                     ; nope
00000000'GF   16    108B  3332         JSB     G^EXE$DEANONPAGED       ; back to the pool
        50    55    D0    1091  3333 10$:     MOVL    R5,R0                   ; Now for the iirp itself
00000000'GF   16    1094  3334         JSB     G^EXE$DEANONPAGED       ; back to the pool
                        109A  3335
              DF    11    109A  3336         BRB     FOUND_EXIT_2            ; Now we are done here
                        109C  3337
```

```
                     109C  3339          .SBTTL  CT_SEND_UNREAD  - Cancel irps
                     109C  3340 ;
                     109C  3341 ; CT_SEND_UNREAD
                     109C  3342 ;
                     109C  3343 ;       Cancel irps by sending a message to the terminal system.
                     109C  3344 ;
                     109C  3345 ; inputs:
                     109C  3346 ;       r2 -> channel
                     109C  3347 ;       r4 -> pcb for process
                     109C  3348 ;       r5 -> CT ucb
                     109C  3349 ;
                     109C  3350
                     109C  3351 CT_SEND_UNREAD:
                     109C  3352
        007C 8F   BB 109C  3353          PUSHR   #^M<R2,R3,R4,R5,R6>
     56 00B8 C5   7E 10A0  3354          MOVAQ   UCB$L_RTT_IRPFL(R5),R6  ; Point to the irp queue
             56   DD 10A5  3355          PUSHL   R6                      ; save its address
                     10A7  3356 ;
                     10A7  3357 ;       20(SP)  R6
                     10A7  3358 ;       16      R5
                     10A7  3359 ;       12      R4
                     10A7  3360 ;       8       R3
                     10A7  3361 ;       4       R2 (channel)
                     10A7  3362 ;       0       IRP LIST HEAD
                     10A7  3363
        56   66   D0 10A7  3364 10$:    MOVL    (R6),R6                 ; Point to next irp
        6E   56   D1 10AA  3365          CMPL    R6,(SP)                 ; End of queue?
             35   13 10AD  3366          BEQL    20$                     ; Yes
  28 A6    04 AE   B1 10AF  3367          CMPW    4(SP),IRP$W_CHAN(R6)    ; Is this the correct channel?
             F1   12 10B4  3368          BNEQ    10$                     ; Nope, try next?
  0C A6    60 A4   D1 10B6  3369          CMPL    PCB$L_PID(R4), -        ; Do the pids match?
                   10BB  3370                  IRP$L_PID(R6)           ;
             EA   12 10BB  3371          BNEQ    10$                     ; Nope, try next
             03   91 10BD  3372          CMPB    #CTP$C_MT_READ_DATA,-
           40 A6       10BF  3373                  IRP$B_CT_RESPTYPE(R6)   ; Is it a read request?
             E4   12 10C1  3374          BNEQ    10$                     ; If no, no way to cancel
           41 A6   95 10C3  3375          TSTB    IRP$B_CT_CANCEL(R6)     ; Did we send a cancel?
             1C   12 10C6  3376          BNEQ    20$                     ; We are done. just return
                   10C8  3377
        51   2C   D0 10C8  3378          MOVL    #CTP$C_UR_LEN,R1        ; Get a message buffer for cancel
           F8B1   30 10CB  3379          BSBW    ALLOC_CTP               ; allocate a buffer
           0F 50  E9 10CE  3380          BLBC    R0,15$                  ; If error, exit
                   10D1  3381 ;
                   10D1  3382 ; Set up unread message
                   10D1  3383 ;
                   10D1  3384          ASSUME  CTP$B_UR_FLAGS EQ CTP$B_MSGTYPE+1
             05   9B 10D1  3385          MOVZBW  #CTP$C_MT_UNREAD,-
           2A A2       10D3  3386                  CTP$B_MSGTYPE(R2)       ;
             06   B0 10D5  3387          MOVW    #CTP$C_UR_MSGLEN,-      ;
           14 A2       10D7  3388                  CTP$W_DATSIZE(R2)       ; Set size of message
             02   B0 10D9  3389          MOVW    #CTP$C_UR_PROLEN,-
           28 A2       10DB  3390                  CTP$W_MSGSIZE(R2)       ; size of protocol with message
            FC7A   30 10DD  3391          BSBW    CT_NET_Q_MSG            ; Send the message
                   10E0  3392
  41 A6    01   90 10E0  3393 15$:    MOVB    #1,IRP$B_CT_CANCEL(R6)  ; Mark for we sent it
                   10E4  3394
             51 8ED0 10E4  3395 20$:    POPL    R1                      ; Discard stack longword to r1
```

```
007C 8F    BA  10E7  3396          POPR    #^M<R2,R3,R4,R5,R6>     ; Restore regs and return
           05  10EB  3397          RSB
               10EC  3398
```

CTD
Pse

PSE

$AB
$$$
$$$
$$$

Pha

Ini
Com
Pas
Sym
Pas
Sym
Pse
Cro
Ass

The
254
The
367
56

Mac

_$2
_$2
_$2
_$2
TOT

413

The

MAC

```
                     10EC  3400 .SBTTL  SEND_UNBIND - send unbind foundation message
                     10EC  3401
                     10EC  3402 ;
                     10EC  3403 ; Input:
                     10EC  3404 ;       R0 - unbind reason code.
                     10EC  3405 ;
                     10EC  3406 SEND_UNBIND:
                     10EC  3407
        50    DD     10EC  3408         PUSHL   R0                          ; save
    51  29    3C     10EE  3409         MOVZWL  #CTP$B_PRO_FILL+2,R1        ; message length
      F88B    30     10F1  3410         BSBW    ALLOC_CTP                   ; allocate packet
     0F 50    E9     10F4  3411         BLBC    R0,20$                      ; exit if failure
        02    90     10F7  3412         MOVB    #PRO$C_UNBIND,-
     26 A2           10F9  3413                 CTP$B_PRO_MSGTYPE(R2)       ; set unbind type
27 A2  6E    B0     10FB  3414         MOVW    (SP),CTP$B_PRO_FILL(R2)     ; set reason for unbind
        03    B0     10FF  3415         MOVW    #<CTP$B_PRO_FILL+2-CTP$B_PRO_MSGTYPE>,-
     14 A2           1101  3416                 CTP$W_DATSIZE(R2)           ; set transfer size
      FC54   30     1103  3417         BSBW    CT_NET_Q_MSG                ; Queue message
                     1106  3418 20$:
        8E   D5     1106  3419         TSTL    (SP)+                       ; get rid of reason code
             05     1108  3420         RSB                                 ; return
                    1109  3421
```

```
                        1109  3423 .SBTTL  STARTNETRCV - initial net startup code
                        1109  3424
                        1109  3425 STARTNETRCV:
                        1109  3426
                        1109  3427 ;
                        1109  3428 ; INIT UCB: (THIS SHOULDN'T BE DONE HERE...) ???
                        1109  3429 ;
00E8 C5   D5   1109  3430         TSTL    UCB$L_CT_NETQFL(R5)     ; Been here before?
      3D   12   110D  3431         BNEQ    10$                     ; Branch if yes
                110F  3432
00E8 C5   9E   110F  3433         MOVAB   UCB$L_CT_NETQFL(R5),-   ; Init queue
00E8 C5                1113  3434                 UCB$L_CT_NETQFL(R5)
00E8 C5   9E   1116  3435         MOVAB   UCB$L_CT_NETQFL(R5),-
00EC C5                111A  3436                 UCB$L_CT_NETQBL(R5)
                111D  3437
00F0 C5   9E   111D  3438         MOVAB   UCB$L_CT_STALLQFL(R5),- ; Init queue
00F0 C5                1121  3439                 UCB$L_CT_STALLQFL(R5)
00F0 C5   9E   1124  3440         MOVAB   UCB$L_CT_STALLQFL(R5),-
00F4 C5                1128  3441                 UCB$L_CT_STALLQBL(R5)
                112B  3442         ;
                112B  3443         ; Initialize some parameters that should be reset by INIT message
                112B  3444         ; from server.
                112B  3445         ;
0106 C5   0200 8F   B0  112B  3446         MOVW    #512,UCB$W_CT_MAXREAD(R5) ; Maximum read buffer in server
          014B 8F   B0  1132  3447         MOVW    #CT$K_MAXCTP,-
          0104 C5         1136  3448                 UCB$W_CT_MAXMSG(R5)     ; Maximum net message
                1139  3449
          011C C5   9E  1139  3450         movab   ucb$l_cnt_frc(r5),-     ; %%% performance hook
          0120 C5         113D  3451                 ucb$l_cnt_addr(r5)      ;
          0118 C5   D4  1140  3452         clrl    ucb$l_cnt_ovr(r5)       ; %%% performance hook
          011C C5   D4  1144  3453         clrl    ucb$l_cnt_frc(r5)       ; %%% performance hook
          0114 C5   D4  1148  3454         clrl    ucb$l_cnt_tqe(r5)       ; %%% performance hook
                114C  3455 10$:
                114C  3456         ;
                114C  3457         ; Allocate TQE
                114C  3458         ;
     00E4 C5   D4  114C  3459         CLRL    UCB$L_CT_TQE(R5)        ; Clear address
                1150  3460
     51   30   9A  1150  3461         MOVZBL  #TQE$C_LENGTH,R1        ; Set length
00000000'GF   16  1153  3462         JSB     G^EXE$ALONONPAGED       ; Allocate the buffer
       25 50   E9  1159  3463         BLBC    R0,15$                  ; Branch if error, forget timer
 0A A2    0F   90  115C  3464         MOVB    #DYN$C_TQE, -
                1160  3465                 TQE$B_TYPE(R2)          ; Set block type
 08 A2    51   B0  1160  3466         MOVW    R1,TQE$W_SIZE(R2)       ; Set size
       10 A2   7C  1164  3467         CLRQ    TQE$L_FR3(R2)           ; Clear flags and ucb address
    F188 CF   9E  1167  3468         MOVAB   W^TQE_WAKEUP,-
       0C A2        116B  3469                 TQE$L_FPC(R2)           ; Set subroutine address
             01   90  116D  3470         MOVB    #TQE$C_SSSNGL,-
       0B A2        116F  3471                 TQE$B_RQTYPE(R2)        ; Set request type as system subroutine
001E8480 8F   D0  1171  3472         MOVL    #CT$K_TQE_DELTA,-
       20 A2        1177  3473                 TQE$Q_DELTA(R2)         ; Set Delta
       24 A2   D4  1179  3474         CLRL    TQE$Q_DELTA+4(R2)       ; Set Delta
00E4 C5    52   D0  117C  3475         MOVL    R2,UCB$L_CT_TQE(R5)     ; Set Timer Queue Entry address
                1181  3476 15$:
                1181  3477         ;
                1181  3478         ; Allocate READ IIRP
                1181  3479         ;
```

```
                00C0 C5   D5  1181  3480          TSTL    UCB$L_CT_RIIRP(R5)      ; Is the iirp already out?
                     64   12  1185  3481          BNEQ    20$                     ; Yes, then ignore it
                   0090   30  1187  3482          BSBW    CT_MAKEIIRP             ; Make an iirp for use
                  5F 50   E9  118A  3483          BLBC    R0, 30$                 ; No good, clean it all up
           00C0 C5    52   D0  118D  3484          MOVL    R2, UCB$L_CT_RIIRP(R5)  ; Save the address of the iirp
        0C A2    FC89 CF   9E  1192  3485          MOVAB   W^CT_NETREADDONE, -     ; Stuff the post address
                           1198  3486                  IRP$L_PID(R2)           ;
             20 A2   21    B0  1198  3487          MOVW    #IO$_READLBLK, -        ; Set the function
                           119C  3488                  IRP$W_FUNC(R2)          ;
                           119C  3489
                      38   BB  119C  3490          PUSHR   #^M<R3,R4,R5>           ; Save
                  53   52   D0  119E  3491          MOVL    R2,R3                   ; Set R3 as IIRP
                55    1C A3   D0  11A1  3492          MOVL    IRP$L_UCB(R3),R5        ; Set the net ucb address up
                      2C A3   D4  11A5  3493          CLRL    IRP$L_SVAPTE(R3)        ; forget it
                00000000'GF   B0  11A8  3494          MOVW    G^IOC$GW_MAXBUF,-       ; setup for another read from net
                      32 A3       11AE  3495                  IRP$W_BCNT(R3)          ; with requested buffer size
                      02   A8  11B0  3496          BISW    #IRP$M_FUNC,-
                      2A A3       11B2  3497                  IRP$W_STS(R3)           ; Set read bit
                00000000'GF   16  11B4  3498          JSB     G^EXE$ALTQUEPKT         ; queue to net driver
                      38   BA  11BA  3499          POPR    #^M<R3,R4,R5>           ; Restore
                           11BC  3500          ;
                           11BC  3501          ; Allocate the write IIRP
                           11BC  3502          ;
                00E0 C5    D5  11BC  3503          TSTL    UCB$L_CT_WIIRP(R5)      ; Already done?
                     29    12  11C0  3504          BNEQ    20$                     ; yes
                   0055    30  11C2  3505          BSBW    CT_MAKEIIRP             ; Make an iirp for use
                  24 50    E9  11C5  3506          BLBC    R0, 30$                 ; No good, clean it all up
           00E0 C5    52   D0  11C8  3507          MOVL    R2, UCB$L_CT_WIIRP(R5)  ; Save the address of the iirp
        0C A2    FBD4 CF   9E  11CD  3508          MOVAB   W^CT_NETWRTDONE, -      ; Stuff the post address
                           11D3  3509                  IRP$L_PID(R2)           ;
                           11D3  3510          ;
                           11D3  3511          ; Determine if we're talking to RTPAD...
                           11D3  3512          ;
     00D9 C5    07   B1   11D3  3513          CMPW    #7,UCB$W_RTT_SYSTYPE(R5) ; VMS?
                05   12   11D8  3514          BNEQ    17$                     ; nope
                20   A8   11DA  3515          BISW    #FLG$M_VAXTOVAX,-
           00DC C5         11DC  3516                  UCB$W_CT_FLAGS(R5)      ; yup, set flag
                           11DF  3517 17$:
           0040 8F    A8   11DF  3518          BISW    #FLG$M_BUFFER,-
           00DC C5         11E3  3519                  UCB$W_CT_FLAGS(R5)      ; Defaults to buffering
                           11E6  3520          ;
                           11E6  3521          ; Send the INIT message
                           11E6  3522          ;
                0A   10    11E6  3523          BSBB    SEND_INIT               ; Send init message *** right place?
              01 50  E9    11E8  3524          BLBC    R0,30$                  ; hangup if error
                           11EB  3525 20$:
                05         11EB  3526          RSB                             ; Return
                           11EC  3527 ;
                           11EC  3528 ;        We are in deep trouble.  Hangup the terminal to run it down
                           11EC  3529 ;        and return failure in r0.  This is done when we cannot obtain
                           11EC  3530 ;        memory for an iirp or any thing else.  IPL can be anything.
                           11EC  3531 ;
                           11EC  3532 ; inputs:
                           11EC  3533 ;        r5 -> CT ucb
                           11EC  3534 ;
                           11EC  3535 30$:
                 F9E7      30    11EC  3536          BSBW    CT_HANGUP               ; Post irps and attn asts
```

```
                50   D4  11EF  3537          CLRL    R0                          ; return failure
                     05  11F1  3538          RSB
                         11F2  3539
                         11F2  3540 SEND_INIT:
              51   49 8F   9A  11F2  3541          MOVZBL  #C_INIT_MSGLEN+CTP$W_MSGSIZE,R1 ; Length of message
                   F786    30  11F6  3542          BSBW    ALLOC_CTP                   ; Allocate net write packet
                 1D 50     E9  11F9  3543          BLBC    R0,10$                      ; exit on error
                    3C     BB  11FC  3544          PUSHR   #^M<R2,R3,R4,R5>            ; Save
                    23     B0  11FE  3545          MOVW    #C_INIT_MSGLEN+2,-
                 14 A2         1200  3546                  CTP$W_DATSIZE(R2)           ; size for net write (+2 for foundation)
            EE91 CF  21     28  1202  3547          MOVC3   #C_INIT_MSGLEN,INIT_MSGBLK,-
                 28 A2         1207  3548                  CTP$W_MSGSIZE(R2)           ; Copy Block
                    3C     BA  1209  3549          POPR    #^M<R2,R3,R4,R5>            ; restore
             00000000'GF   B0  120B  3550          MOVW    G^IOC$GW_MAXBUF,-
                 39 A2         1211  3551                  W_INIT_BUFSIZ(R2)           ; set maxbuf
                               1213  3552          ;
                               1213  3553          ; write to net
                               1213  3554          ;
                   FB44    30  1213  3555          BSBW    CT_NET_Q_MSG                ; write to net
                50   01    D0  1216  3556          MOVL    #1,R0
                               1219  3557 10$:
                         05    1219  3558          RSB                                 ; returrn
                               121A  3559
                               121A  3560
```

```
                     121A  3562          .SBTTL CT_MAKEIIRP  - Manufacture an internal irp
                     121A  3563 ;
                     121A  3564 ; CT_MAKEIIRP
                     121A  3565 ;
                     121A  3566 ;       Make an internal IRP for sending to the netdriver.
                     121A  3567 ;       If we can't get the space, return failure.
                     121A  3568 ;
                     121A  3569 ; inputs:
                     121A  3570 ;       r3 -> CT irp
                     121A  3571 ;       r5 -> CT ucb
                     121A  3572 ;
                     121A  3573 ; outputs:
                     121A  3574 ;       r0 = success or fail
                     121A  3575 ;
                     121A  3576
                     121A  3577 CT_MAKEIIRP:
                     121A  3578
 51    C4 8F    9A   121A  3579          MOVZBL  #IRP$C_LENGTH,R1        ; Obtain a buffer of correct size
00000000'GF    16   121E  3580          JSB     G^EXE$ALONONPAGED       ; from dynamic memory
      3A 50    E9   1224  3581          BLBC    R0,10$                  ; No memory left, so return error
 0A A2    0A    90   1227  3582          MOVB    #DYN$C_IRP,-            ; Set the type and size fields
                     122B  3583                  IRP$B_TYPE(R2)          ;
 08 A2    51    B0   122B  3584          MOVW    R1,IRP$W_SIZE(R2)       ;
    0C A2       D4   122F  3585          CLRL    IRP$L_PID(R2)           ; No pid here
 10 A2    55    D0   1232  3586          MOVL    R5,IRP$L_AST(R2)        ; Save the CT ucb field
    00B4 C5    D0   1236  3587          MOVL    UCB$L_RTT_NETWIND(R5),- ; Set up the window
      18 A2         123A  3588                  IRP$L_WIND(R2)          ;
    00B0 C5    D0   123C  3589          MOVL    UCB$L_RTT_NETUCB(R5),-  ; and the ucb for the net
      1C A2         1240  3590                  IRP$L_UCB(R2)           ;
          20   B0   1242  3591          MOVW    #IO$_WRITELBLK,-        ; the function
    20 A2         1244  3592                  IRP$W_FUNC(R2)          ;
23 A2    04    90   1246  3593          MOVB    #4,IRP$B_PRI(R2)        ; priority of this in queue
       01    B0   124A  3594          MOVW    #IRP$M_BUFIO,-          ; Its a buffered io function
    2A A2         124C  3595                  IRP$W_STS(R2)           ; and assume a write
    30 A2    B4   124E  3596          CLRW    IRP$W_BOFF(R2)          ; no quota to return for iirp
    38 A2    7C   1251  3597          CLRQ    IRP$L_IOST1(R2)         ; no status yet
                  1254  3598          ASSUME  IRP$L_OBCNT -
                  1254  3599                  EQ -
                  1254  3600                  IRP$L_ABCNT+4
    40 A2    7C   1254  3601          CLRQ    IRP$L_ABCNT(R2)         ; Some more byte counts
    50 A3    D0   1257  3602          MOVL    IRP$L_SEQNUM(R3),-      ; Grab a quick sequence number
    50 A2         125A  3603                  IRP$L_SEQNUM(R2)        ;
    58 A3    D0   125C  3604          MOVL    IRP$L_ARB(R3),-         ; Access rights block, incase needed
    58 A2         125F  3605                  IRP$L_ARB(R2)           ;
             05   1261  3606 10$:     RSB
                  1262  3607
```

```
1262  3609 .IF DF DEBUG_LOG
1262  3610 .sbttl TTY$LOG_IO - log data to mailbox
1262  3611 ;
1262  3612 ; R5 - RT UCB
1262  3613 ; R6 - message length
1262  3614 ; R7 - message address
1262  3615 ; R8 - message type
1262  3616 ;
1262  3617
1262  3618 tty$log_io:
1262  3619
1262  3620         BBS     #1,G^SGN$GL_VMSD4,5$    ; branch if all terminals logging
1262  3621         BBS     #tt2$v_disconnect,-
1262  3622                 ucb$l_devdepnd2(R5),5$  ; or this terminal set for logging
1262  3623         rsb
1262  3624 5$:
1262  3625         pushr   #^m<R0,r1,r2,r3,r4,r5>
1262  3626
1262  3627         movl    G^sgn$gl_vms8,R0
1262  3628         beql    100$                    ; exit if 0
1262  3629         cmpb    #dyn$c_ucb,ucb$b_type(R0)
1262  3630         bneq    100$                    ; exit if not ucb
1262  3631
1262  3632         addl3   #6+12,R6,R1             ; message length + non-paged header
1262  3633         jsb     G^exe$alononpaged       ; get buffer
1262  3634         blbc    r0,100$                 ; exit if no memory
1262  3635         MOVB    #DYN$C_UNUSED_2, -
1262  3636                 TQE$B_TYPE(R2)          ; Set block type
1262  3637         MOVW    R1,TQE$W_SIZE(R2)       ; Set size
1262  3638
1262  3639         pushl   R2                      ; save address
1262  3640         movab   12(R2),R2               ; Skip non-paged header
1262  3641         pushl   R2                      ; save
1262  3642         movw    r8,(r2)+                ; message type
1262  3643         movw    ucb$w_unit(R5),(R2)+    ; unit #
1262  3644         movw    r6,(R2)+                ; message length
1262  3645         beql    10$                     ; branch if zero
1262  3646         movc3   r6,(r7),(r2)            ; data
1262  3647 ;
1262  3648 ; send message
1262  3649 ;
1262  3650 10$:
1262  3651         addl3   #6,R6,R3                ; length
1262  3652         popl    R4                      ; address of message
1262  3653         movl    G^sgn$gl_vms8,R5        ; mailbox ucb
1262  3654         jsb     G^exe$wrtmailbox        ; write to mailbox
1262  3655
1262  3656 99$:    popl    r0                      ; restore address
1262  3657         jsb     G^exe$deanonpaged
1262  3658
1262  3659 100$:   popr    #^m<R0,r1,r2,r3,r4,r5>
1262  3660         rsb
1262  3661 .endc
1262  3662
```

```
                                     J 2
     1262  3664          .SBTTL  CT_END, End of driver
     1262  3665
     1262  3666 ;
     1262  3667 ; Label that marks the end of the driver
     1262  3668 ;
     1262  3669
00000000   3670 .PSECT  $$$116_DRIVER,LONG
     0000  3671 CT_END:
     0000  3672          .END
```

```
$$$                             = 00000020 R      02
$$OP                            = 00000002
ACB$L_KAST                      = 00000018
ALLOC_CTP                         0000097F R      03
ALLOC_ERR                         00000979 R      03
ALTECHADDR                        00000020
ALTECHSIZE                        00000024
AQB$L_ACPPID                    = 0000000C
AST_ABORT                       = 00000008
AST_EXCLUDE                     = 00000004
AST_INCLUDE                     = 00000000
AT$_NULL                        = 00000005
BIT...                          = 00000002
BUFADDR                           00000000
BUFSIZE                           00000004
BUG$_CTERM                        ********    X   03
CAN$C_CANCEL                    = 00000000
CAN$C_DASSGN                    = 00000001
CANCEL_LIST                       00000B98 R      03
CAN_CTRLO                         000000CF R      03
CH$C_CTERM                      = 00000002
CH$C_CT_CHAR_ATT                = 00000002
CH$C_CT_INPUT_COUNT             = 00000008
CHECK_POST_IO                     00000AED R      03
COM$DELATTNASTP                   ********    X   03
COM$DELCTRLASTP                   ********    X   03
COM$DRVDEALMEM                    ********    X   03
COM$FLUSHATTNS                    ********    X   03
COM$FLUSHCTRLS                    ********    X   03
COM$POST                          ********    X   03
COM$SETATTNAST                    ********    X   03
COM$SETCTRLAST                    ********    X   03
CRB$L_INTD                      = 00000024
CT$DDT                            00000000 RG     03
CT$K_CTPLOLIM                   = 00000001
CT$K_CTPQLIM                    = 00000003
CT$K_FIPL                       = 00000008
CT$K_MAXCTP                     = 0000014B
CT$K_TQE_DELTA                  = 001E8480
CTERM_DISCARD                     00000FC9 R      03
CTERM_INIT                        00000EDC R      03
CTERM_IN_STATE                    00000FAD R      03
CTERM_OUTBAND                     00000FE0 R      03
CTP$AB_SENSEBUF                   ********    X   03
CTP$B_CH_FLAGS                  = 0000002B
CTP$B_CI_FLAGS                  = 0000002B
CTP$B_DS_FLAGS                  = 0000002B
CTP$B_IN_ECO                    = 0000002D
CTP$B_IN_PARMTYPE               = 00000037
CTP$B_IN_VERSION                = 0000002C
CTP$B_IS_FLAGS                  = 0000002B
CTP$B_MSGTYPE                   = 0000002A
CTP$B_OB_CHAR                   = 0000002C
CTP$B_PRO_FILL                  = 00000027
CTP$B_PRO_MSGTYPE               = 00000026
CTP$B_RD_CURS_POS               = 0000002F
CTP$B_RD_FLAGS                  = 0000002B
CTP$B_TYPE                      = 0000000A
CTP$B_UR_FLAGS                  = 0000002B
CTP$B_WC_FLAGS                  = 0000002B
CTP$B_WR_POSTFIX                = 0000002E
CTP$B_WR_PREFIX                 = 0000002D
CTP$C_CH_CANCEL                 = 00000000
CTP$C_CH_ECHONONE               = 00000000
CTP$C_CH_ECHOSTANDARD           = 00000002
CTP$C_CH_HELLO                  = 00000003
CTP$C_CH_ICLEAR                 = 00000001
CTP$C_CH_PROLEN                 = 00000004
CTP$C_CI_LEN                    = 0000002C
CTP$C_IC_MSGLEN                 = 00000008
CTP$C_MT_CHAR                   = 0000000B
CTP$C_MT_CHECK_INP              = 0000000C
CTP$C_MT_INIT                   = 00000001
CTP$C_MT_INP_COUNT              = 0000000D
CTP$C_MT_READ_DATA              = 00000003
CTP$C_MT_START_RD               = 00000002
CTP$C_MT_UNREAD                 = 00000010
CTP$C_MT_VMSQIO                 = 0000000F
CTP$C_MT_VMS_READVFY            = 00000011
CTP$C_MT_WRITE                  = 00000007
CTP$C_MT_WRITE_COM              = 00000008
CTP$C_SR2_LEN                   = 00000043
CTP$C_SR_LEN                    = 0000003B
CTP$C_SR_THISTERM               = 00000001
CTP$C_UR_LEN                    = 0000002C
CTP$C_UR_MSGLEN                 = 00000006
CTP$C_UR_PROLEN                 = 00000002
CTP$C_WR_CHAR                   = 00000002
CTP$C_WR_LEN                    = 0000002F
CTP$C_WR_NEWLINECNT             = 00000001
CTP$K_SENSEBUF                    ********    X   03
CTP$L_MSGDAT                    = 0000000C
CTP$L_SR_FLAGS                  = 0000002B
CTP$L_USRBFR                    = 00000010
CTP$M_CH_D                      = 00000008
CTP$M_CH_EE                     = 00000030
CTP$M_CH_I                      = 00000004
CTP$M_CH_OO                     = 00000003
CTP$M_RD_INPFULL                = 00000004
CTP$M_SR_ALLBUTX                = 00000300
CTP$M_SR_EDIT                   = 0000020C
CTP$M_SR_ESCAPE                 = 00020000
CTP$M_SR_FORMAT                 = 00000008
CTP$M_SR_LOWTOUP                = 00000080
CTP$M_SR_NOECHO                 = 00000800
CTP$M_SR_NORMTERM               = 00008000
CTP$M_SR_PURGE                  = 00000004
CTP$M_SR_TIMED                  = 00002000
CTP$M_SR_TRMECHO                = 00001000
CTP$M_WR_BEFAFT                 = 00000002
CTP$M_WR_BEFAFTRE               = 00000003
CTP$M_WR_BEGIN                  = 00000010
CTP$M_WR_DISCARD                = 00000008
CTP$M_WR_END                    = 00000020
```

```
CTPSM_WR_NEWLINE        =  00000004
CTPSM_WR_STATUS         =  00000400
CTPSM_WR_TRANSPARENT    =  00000800
CTPSS_SR_TERM_SET       =  00000002
CTPST_RD_DATA           =  00000032
CTPST_SR2_TERM          =  00000042
CTPST_SR_TERM           =  0000003A
CTPST_WR_DATA           =  0000002F
CTPSV_CH_EE             =  00000004
CTPSV_DS_DISCARD        =  00000000
CTPSV_IS_NONZERO        =  00000000
CTPSV_SR_NOEDIT         =  00000012
CTPSV_SR_TERM_SET       =  0000000E
CTPSV_WC_DISCARD        =  00000000
CTPSV_WR_POSTFIX        =  00000008
CTPSV_WR_PREFIX         =  00000006
CTPSW_CH_PARAM          =  0000002C
CTPSW_DATSIZE           =  00000014
CTPSW_IC_COUNT          =  0000002C
CTPSW_MSGSIZE           =  00000028
CTPSW_RD_TERM_POS       =  00000030
CTPSW_SIZE              =  00000008
CTPSW_SR2_ALTECHSIZE    =  0000003A
CTPSW_SR2_EDITFLAGS     =  0000003E
CTPSW_SR2_FILLCHAR      =  00000040
CTPSW_SR2_PICSTRSIZE    =  0000003C
CTPSW_SR_END_DATA       =  00000030
CTPSW_SR_END_PRMT       =  00000034
CTPSW_SR_LO_WATER       =  00000038
CTPSW_SR_MAX_LEN        =  0000002E
CTPSW_SR_STR_DISP       =  00000036
CTPSW_SR_TIMEOUT        =  00000032
CTPSW_WC_HORPOS         =  0000002C
CTPSW_WC_VERPOS         =  0000002E
CTPSW_WR_FLAGS          =  0000002B
CTRLO                     000000B9 R      03
CTRL_CY                   000006E2 R      03
CT_ABORTIRPS              00000C0E R      03
CT_CANCEL                 00000B03 R      03
CT_END                    00000000 R      04
CT_FUNCTABLE              00000038 R      03
CT_HANGUP                 00000BD6 R      03
CT_INTERRUPT              0000098F R      03
CT_MAKEIIRP               0000121A R      03
CT_NETMSGSENDX            00000D17 R      03
CT_NETREADDONE            00000E1F R      03
CT_NETWRTDONE             00000DA5 R      03
CT_NET_Q_MSG              00000D5A R      03
CT_NET_WRITE              00000D72 R      03
CT_POST_SENSE             ********    X   03
CT_READ                   00000346 R      03
CT_READ_ITMLST            000004D1 R      03
CT_SEND_UNREAD            0000109C R      03
CT_SENSEMODE              00000862 R      03
CT_SET                    ********    X   03
CT_SETMODE                0000060D R      03
CT_UNSOLIC                00000E07 R      03
CT_VMSQIO_MSG             ********    X   03
CT_VMS_BRDCST             ********    X   03
CT_VMS_SENSEMODE          ********    X   03
CT_VMS_SETMODE            ********    X   03
CT_WRITE                  000000C6 R      03
CT_WRITE_FILLIN           00000277 R      03
CT_WRITE_WRTCTP           00000CE0 R      03
C_INIT_MSGLEN           =  00000021
DCS_TERM                =  00000042
DDB$L_ACPD              =  00000010
DDB$L_DDT               =  0000000C
DEV$M_AVL               =  00040000
DEV$M_CCL               =  00000002
DEV$M_IDV               =  04000000
DEV$M_ODV               =  08000000
DEV$M_REC               =  00000001
DEV$M_RTT               =  00000004
DEV$M_TRM               =  00000004
DEV$V_DMT               =  00000015
DPT$C_LENGTH            =  00000038
DPT$C_VERSION           =  00000004
DPT$INITAB                00000038 R      02
DPT$REINITAB              00000086 R      02
DPT$TAB                   00000000 R      02
DYN$C_BUFIO             =  00000013
DYN$C_CRB               =  00000005
DYN$C_DDB               =  00000006
DYN$C_DPT               =  0000001E
DYN$C_IRP               =  0000000A
DYN$C_ORB               =  00000049
DYN$C_TQE               =  0000000F
DYN$C_UCB               =  00000010
EDITFLAGS                  00000038
EDITMODE                   0000003C
EXE$ABORTIO               ********    X   03
EXE$ALLOCBUF              ********    X   03
EXE$ALONONPAGED           ********    X   03
EXE$ALTQUEPKT             ********    X   03
EXE$AL_TQENOREPT          ********    X   03
EXE$BUFFRQUOTA            ********    X   03
EXE$CARRIAGE              ********    X   03
EXE$DEANONPAGED           ********    X   03
EXE$FINISHIOC             ********    X   03
EXE$GQ_SYSTIME            ********    X   03
EXE$INSERTIRP             ********    X   03
EXE$INSTIMQ               ********    X   03
EXE$MAXACMODE             ********    X   03
EXE$PROBER                ********    X   03
EXE$QIORETURN             ********    X   03
EXE$READCHK               ********    X   03
EXE$SNDEVMSG              ********    X   03
EXE$WRITECHK              ********    X   03
FDT_ABORT                 00000928 R      03
FDT_ALLOC_MESSAGE         00000939 RG     03
FDT_CHARSIZE              000008FB R      03
FDT_FINISHIOC             00000918 R      03
FDT_FINISHIOC_OK          00000915 R      03
```

```
FDT_READFLAGS                   000005AD R      03
FDT_SET_OUTBAND                 000007B5 R      03
FETCH_OOB_DATA                  0000081B R      03
FILLCHAR                        0000003A
FLG$M_BUFFER                =   00000040
FLG$M_CANCTRLO              =   00000004
FLG$M_CTRLC                 =   00000080
FLG$M_CTRLO                 =   00000002
FLG$M_INWRTFDT              =   00000008
FLG$M_VAXTOVAX              =   00000020
FLG$M_WIIRP_BSY             =   00000001
FLG$V_BUFFER                =   00000006
FLG$V_CANCTRLO              =   00000002
FLG$V_CTRLC                 =   00000007
FLG$V_CTRLO                 =   00000001
FLG$V_INWRTFDT              =   00000003
FLG$V_VAXTOVAX              =   00000005
FLG$V_WIIRP_BSY             =   00000000
FOUND_ACCEPT                    00000E83 R      03
FOUND_BIND                      00000E77 R      03
FOUND_CONF_MODE                 00000E83 R      03
FOUND_CTERM                     00000E86 R      03
FOUND_ENTR_MODE                 00000E83 R      03
FOUND_ERROR                     0000107F R      03
FOUND_ERROR_2                   00001082 R      03
FOUND_EXIT                      00001057 R      03
FOUND_EXIT_2                    0000107B R      03
FOUND_EXIT_MODE                 00000E83 R      03
FOUND_MODE                      00000E83 R      03
FOUND_NO_MODE                   00000E83 R      03
FOUND_REBIND                    00000E83 R      03
FOUND_UNBIND                    00000E7D R      03
FUNCTAB_LEN                 =   00000040
GET_PARAMS                      000008EA R      03
HANGUP                          00000E7D R      03
INIADDR                         00000018
INIOFFSET                       00000034
INISIZE                         0000001C
INIT_MSGBLK                     00000098 R      03
IO$M_CANCTRLO               =   00000040
IO$M_CVTLOW                 =   00000100
IO$M_NEWLINE                =   00000400
IO$M_NOECHO                 =   00000040
IO$M_NOFILTR                =   00000200
IO$M_NOFORMAT               =   00000100
IO$M_PURGE                  =   00000800
IO$M_REFRESH                =   00002000
IO$M_TIMED                  =   00000080
IO$M_TRMNOECHO              =   00001000
IO$V_BRDCST                 =   0000000E
IO$V_CANCTRLO               =   00000006
IO$V_CVTLOW                 =   00000008
IO$V_ESCAPE                 =   0000000E
IO$V_EXTEND                 =   0000000F
IO$V_MAINT                  =   00000006
IO$V_NEWLINE                =   0000000A
IO$V_NOECHO                 =   00000006
IO$V_NOFILTR                =   00000009
IO$V_NOFORMAT               =   00000008
IO$V_PURGE                  =   0000000B
IO$V_RD_MODEM               =   00000007
IO$V_REFRESH                =   0000000D
IO$V_TIMED                  =   00000007
IO$V_TRMNOECHO              =   0000000C
IO$V_TYPEAHDCNT             =   00000006
IO$_READLBLK                =   00000021
IO$_READPBLK                =   0000000C
IO$_READPROMPT              =   00000037
IO$_READVBLK                =   00000031
IO$_SENSECHAR               =   0000001B
IO$_SENSEMODE               =   00000027
IO$_SETCHAR                 =   0000001A
IO$_SETMODE                 =   00000023
IO$_TTYREADALL              =   0000003A
IO$_TTYREADPALL             =   0000003B
IO$_VIRTUAL                 =   0000003F
IO$_WRITELBLK               =   00000020
IO$_WRITEPBLK               =   0000000B
IO$_WRITEVBLK               =   00000030
IOC$GW_MAXBUF                   ********    X   03
IOC$MNTVER                      ********    X   03
IOC$RETURN                      ********    X   03
IPL$_TIMER                  =   00000008
IRP$B_CARCON                =   0000003C
IRP$B_CT_CANCEL             =   00000041
IRP$B_CT_RESPTYPE           =   00000040  G
IRP$B_PRI                   =   00000023
IRP$B_TYPE                  =   0000000A
IRP$C_LENGTH                =   000000C4
IRP$L_ABCNT                 =   00000040
IRP$L_ARB                   =   00000058
IRP$L_AST                   =   00000010
IRP$L_FPC                   =   0000006C
IRP$L_FR4                   =   00000074
IRP$L_IOST1                 =   00000038
IRP$L_IOST2                 =   0000003C
IRP$L_MEDIA                 =   00000038
IRP$L_OBCNT                 =   00000044
IRP$L_PID                   =   0000000C
IRP$L_SEQNUM                =   00000050
IRP$L_SVAPTE                =   0000002C
IRP$L_UCB                   =   0000001C
IRP$L_WIND                  =   00000018
IRP$M_BUFIO                 =   00000001
IRP$M_FCODE                 =   0000003F
IRP$M_FUNC                  =   00000002
IRP$M_TERMIO                =   00000200
IRP$Q_TT_STATE              =   00000040
IRP$W_BCNT                  =   00000032
IRP$W_BOFF                  =   00000030
IRP$W_CHAN                  =   00000028
IRP$W_CT_POST               =   00000042
IRP$W_FUNC                  =   00000020
IRP$W_SIZE                  =   00000008
```

```
IRP$W_STS                       = 0000002A
JIB$L_BYTCNT                    = 00000020
MASKH                           = 00000008
MASKL                           = 04000000
MSG$_TRMHANGUP                  = 00000006
MSG$_TRMUNSOLIC                 = 00000001
ORB$B_FLAGS                     = 0000000B
ORB$L_OWNER                     = 00000000
ORB$M_PROT_16                   = 00000001
ORB$W_PROT                      = 00000018
P1                              = 00000000
P2                              = 00000004
P3                              = 00000008
P4                              = 0000000C
P5                              = 00000010
P6                              = 00000014
PCB$L_JIB                       = 00000080
PCB$L_PID                       = 00000060
PICSTRADDR                        00000028
PICSTRSIZE                        0000002C
POST                              00000AC9 R      03
POST_COUNT                      = 00000007
POST_READ                         000009D0 R      03
POST_SENSE                        00000A42 R      03
POST_SENSE_TYPEAHD                00000AB1 R      03
POST_WRITE                        00000A95 R      03
PR$_IPL                         = 00000012
PRMADDR                           00000008
PRMSIZE                           0000000C
PRO$C_DATA                      = 00000009
PRO$C_UNBIND                    = 00000002
Q_ERR                             00000D96 R      03
READ_LOCAL                      = 0000003E
REMOTE_1                        = FF28DFFA
REMOTE_2                        = 00000002
SCH$WAKE                          ********    X   03
SEND_INIT                         000011F2 R      03
SEND_OOB_MSG                      0000076F R      03
SEND_UNBIND                       000010EC R      03
SENSE_SPAWN                       00000AD9 R      03
SETUP_OUTBAND                     000007CE R      03
SETUP_TQE                         000002B6 R      03
SET_BRDCST                        000007A3 R      03
SET_CONNECT                       0000079B R      03
SET_CTRLC                         000006D4 R      03
SET_CTRLY                         00000695 R      03
SET_DISCONNECT                    0000079B R      03
SET_HANGUP                        000007B9 R      03
SET_MAINT                         0000079B R      03
SET_MSGHDR                        00000964 R      03
SET_NOP                           00000798 R      03
SET_OUTBAND                       000006F6 R      03
SET_PID                           000007AD R      03
SET_READY                         000007C1 R      03
SIZ...                          = 00000001
SS$_ABORT                       = 0000002C
SS$_ACCVIO                      = 0000000C
SS$_BADPARAM                    = 00000014
SS$_CONTROLC                    = 00000651
SS$_CONTROLO                    = 00000609
SS$_CONTROLY                    = 00000611
SS$_DATAOVERUN                  = 00000838
SS$_DEVREQERR                   = 00000334
SS$_HANGUP                      = 000002CC
SS$_ILLIOFUNC                   = 000000F4
SS$_LINKABORT                   = 000020E4
SS$_NORMAL                      = 00000001
SS$_OPINCOMPL                   = 000002D4
SS$_PARITY                      = 000001F4
SS$_PARTESCAPE                  = 000001FC
SS$_TIMEOUT                     = 0000022C
STARTNETRCV                       00001109 R      03
STAT_TABLE                        00000078 R      03
TAST$B_CTRL                     = 0000002D
TAST$L_FLINK                    = 0000001C
TAST$L_MASK                     = 00000030
TAST$V_ABORT                    = 00000005
TAST$V_INCLUDE                  = 00000001
TAST$V_LOST                     = 00000004
TIMEOUT                           00000030
TQE$B_RQTYPE                    = 0000000B
TQE$B_TYPE                      = 0000000A
TQE$C_LENGTH                    = 00000030
TQE$C_SSREPT                    = 00000005
TQE$C_SSSNGL                    = 00000001
TQE$L_FPC                       = 0000000C
TQE$L_FR3                       = 00000010
TQE$L_FR4                       = 00000014
TQE$M_BSY                       = 00000001
TQE$M_DELETE                    = 00000002
TQE$Q_DELTA                     = 00000020
TQE$S_BSY                       = 00000001
TQE$S_DELETE                    = 00000001
TQE$V_BSY                       = 00000000
TQE$V_DELETE                    = 00000001
TQE$W_SIZE                      = 00000008
TQE_WAKEUP                        000002F6 R      03
TRM$V_TM_AUTO_TAB               = 00000012
TRM$V_TM_NOEDIT                 = 0000000F
TRM$V_TM_NORECALL               = 00000010
TRM$V_TM_R_JUST                 = 00000011
TRMADDR                           00000010
TRMSIZE                           00000014
TT$M_CRFILL                     = 00000400
TT$M_EIGHTBIT                   = 00008000
TT$M_ESCAPE                     = 00000008
TT$M_HOLDSCREEN                 = 00004000
TT$M_HOSTSYNC                   = 00000010
TT$M_LFFILL                     = 00000800
TT$M_LOWER                      = 00000080
TT$M_MECHFORM                   = 00080000
TT$M_MECHTAB                    = 00000100
TT$M_MODEM                      = 00200000
TT$M_NOECHO                     = 00000002
```

```
TT$M_PAGE                       = FF000000
TT$M_SCOPE                      = 00001000
TT$M_SCRIPT                     = 00000040
TT$M_TTSYNC                     = 00000020
TT$M_WRAP                       = 00000200
TT$_UNKNOWN                     = 00000000
TT2$M_AUTOBAUD                  = 00000002
TT2$M_DCL_MAILBX                = 00000200
TT2$M_SYSPWD                    = 00080000
TT2$V_SYSPWD                    = 00000013
TTY$C_CTRLC                     = 00000003
TTY$C_CTRLY                     = 00000019
TTY$C_LF                        = 0000000A
TTY$GL_DEFCHAR                    ********    X  02
TTY$GL_DEFCHAR2                   ********    X  03
TTY$GL_JOBCTLMB                   ********    X  03
TTY$GL_OWNUIC                     ********    X  02
TTY$GW_DEFBUF                     ********    X  02
TTY$GW_PROT                       ********    X  02
UCB$B_CT_CRFILL                   00000110
UCB$B_CT_ECO                    = 0000010D
UCB$B_CT_LFFILL                   00000111
UCB$B_CT_VERSION                = 0000010C
UCB$B_DEVCLASS                  = 00000040
UCB$B_DEVTYPE                   = 00000041
UCB$B_DIPL                      = 0000005E
UCB$B_FIPL                      = 0000000B
UCB$K_RTT_LENGTH                = 00000138
UCB$L_AMB                       = 00000060
UCB$L_CNT_ADDR                    00000120
UCB$L_CNT_FRC                     0000011C
UCB$L_CNT_OVR                     00000118
UCB$L_CNT_TQE                     00000114
UCB$L_CT_ABORT                    00000130
UCB$L_CT_EXCLUDE                  0000012C
UCB$L_CT_INCLUDE                  00000128
UCB$L_CT_LEGALMSG               = 00000108
UCB$L_CT_NETQBL                 = 000000EC
UCB$L_CT_NETQFL                 = 000000E8
UCB$L_CT_PID                      00000124
UCB$L_CT_RIIRP                  = 000000C0
UCB$L_CT_STALLQBL               = 000000F4
UCB$L_CT_STALLQFL               = 000000F0
UCB$L_CT_TQE                    = 000000E4
UCB$L_CT_WIIRP                  = 000000E0
UCB$L_CT_WRTCTP                 = 000000F8
UCB$L_CT_WRTCUR                 = 000000FC
UCB$L_DEVCHAR                   = 00000038
UCB$L_DEVCHAR2                  = 0000003C
UCB$L_DEVDEPEND                 = 00000044
UCB$L_DEVDEPND2                 = 00000048
UCB$L_RTT_IRPBL                 = 000000BC
UCB$L_RTT_IRPFL                 = 000000B8
UCB$L_RTT_NETIRP                = 000000C0
UCB$L_RTT_NETUCB                = 000000B0
UCB$L_RTT_NETWIND               = 000000B4
UCB$L_TL_BANDQUE                = 0000009C
UCB$L_TL_CTLPID                 = 000000A4
UCB$L_TL_CTRLC                  = 00000094
UCB$L_TL_CTRLY                  = 00000090
UCB$L_TL_OUTBAND                = 00000098
UCB$L_VCB                       = 00000034
UCB$M_JOB                       = 00000001
UCB$M_TT_HANGUP                 = 00000008
UCB$M_VALID                     = 00000800
UCB$Q_TL_BRKTHRU                = 000000A8
UCB$T_CT_DEBUG_FILL             = 00000110
UCB$V_JOB                       = 00000000
UCB$V_ONLINE                    = 00000004
UCB$V_TT_HANGUP                 = 00000003
UCB$W_CT_FLAGS                  = 000000DC
UCB$W_CT_MAXMSG                 = 00000104
UCB$W_CT_MAXREAD                = 00000106
UCB$W_CT_PARITY                   00000134
UCB$W_CT_QCTPCNT                = 000000DE
UCB$W_CT_SPEED                    00000112
UCB$W_CT_WRTCNT                 = 00000102
UCB$W_CT_WRTSIZ                 = 00000100
UCB$W_DEVBUFSIZ                 = 00000042
UCB$W_DEVSTS                    = 00000068
UCB$W_ERRCNT                    = 00000082
UCB$W_REFC                      = 0000005C
UCB$W_RTT_SYSTYPE               = 000000D9
UCB$W_STS                       = 00000064
UNBIND$C_DISCONNECT             = 00000004
UNBIND$C_USER                   = 00000003
VCB$L_AQB                       = 00000010
VMS_BRDCST                        00001054 R      03
VMS_QIO                           0000104C R      03
W_INIT_BUFSIZ                   = 00000039
```

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	00000138 (312.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$105_PROLOGUE	00000091 (145.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	00001262 (4706.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$116_DRIVER	00000000 (0.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.06	00:00:00.42
Command processing	129	00:00:00.54	00:00:01.81
Pass 1	929	00:00:29.58	00:01:00.41
Symbol table sort	0	00:00:04.31	00:00:09.13
Pass 2	410	00:00:07.41	00:00:16.00
Symbol table output	1	00:00:00.29	00:00:00.45
Psect synopsis output	0	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1500	00:00:42.21	00:01:28.24

The working set limit was 3000 pages.
254621 bytes (498 pages) of virtual memory were used to buffer the intermediate code.
There were 220 pages of symbol table space allocated to hold 3925 non-local and 204 local symbols.
3672 source lines were read in Pass 1, producing 31 object records in Pass 2.
56 pages of virtual memory were used to define 53 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]REM.MLB;1	0
_255DUA28:[RTPAD.OBJ]RTPAD.MLB;1	2
_255DUA28:[SYS.OBJ]LIB.MLB;1	32
_255DUA28:[SYSLIB]STARLFT.MLB;2	16
TOTALS (all libraries)	50

4134 GETS were required to define 50 macros.

There were no errors, warnings or information messages.

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

RPGMSGTXT
LIS

DTE_DF03
MAP

RPGMOVE3
LIS

RPGSORT
LIS

RPGOPEN
LIS

RTPAD

CTDRIVER
MAP

RTPAD
MAP

RTPADMACS
MAR

RPGMSGPTR
LIS

RPGVECTOR
LIS

RPGPRINT
LIS

RPGUDATE
LIS

RTDEF
SDL

DTE_DF03
MAR

CTDRIVER
LIS

CTDSENSE
LIS

CTDVMSPEC
LIS

CTSENSERT
LIS

RSTSRT
LIS

CTERMRT
LIS

DTE_DF03
LIS

CTSETRT
LIS

CTDSET
LIS