```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
```

```
RRRRRRRR    TTTTTTTTTT  TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
RRRRRRRR    TTTTTTTTTT  TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
RR      RR      TT          TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
RR      RR      TT          TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
RR      RR      TT          TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
RR      RR      TT          TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
RRRRRRRR        TT          TT      DD      DD  RRRRRRRR      II      VV      VV  EEEEEEEE    RRRRRRRR
RRRRRRRR        TT          TT      DD      DD  RRRRRRRR      II      VV      VV  EEEEEEEE    RRRRRRRR
RR  RR          TT          TT      DD      DD  RR  RR        II      VV      VV  EE          RR  RR
RR  RR          TT          TT      DD      DD  RR  RR        II      VV      VV  EE          RR  RR
RR    RR        TT          TT      DD      DD  RR    RR      II        VV  VV    EE          RR    RR        ....
RR    RR        TT          TT      DD      DD  RR    RR      II        VV  VV    EE          RR    RR        ....
RR      RR      TT          TT      DDDDDDDD    RR      RR    IIIIII      VV      EEEEEEEEEE  RR      RR      ....
RR      RR      TT          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  RTTDRIVER - Remote Terminal Driver
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;*****************************************************************************
0000     5  ;*                                                                           *
0000     6  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8  ;*  ALL RIGHTS RESERVED.                                                     *
0000     9  ;*                                                                           *
0000    10  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    11  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    12  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    13  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    14  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    15  ;*  TRANSFERRED.                                                             *
0000    16  ;*                                                                           *
0000    17  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    18  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    19  ;*  CORPORATION.                                                             *
0000    20  ;*                                                                           *
0000    21  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    22  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23  ;*                                                                           *
0000    24  ;*                                                                           *
0000    25  ;*****************************************************************************
0000    26  ;
0000    27  ;++
0000    28  ;
0000    29  ; FACILITY:
0000    30  ;
0000    31  ;       VAX/VMS Remote Terminal Driver
0000    32  ;
0000    33  ; ABSTRACT:
0000    34  ;
0000    35  ;       This module contains the remote terminal driver routines.  This driver
0000    36  ;       is used by the application process side of the operation.  In other
0000    37  ;       words, it receives the QIO requests from the process that does not
0000    38  ;       have local access to the terminal.
0000    39  ;
0000    40  ;       This driver's primary function is to receive QIO system service
0000    41  ;       requests, repackage the QIO arguments, and hand the new package to
0000    42  ;       the tranport mechanism for delivery to the remote terminal
0000    43  ;       handler process on the system with local access to the terminal.
0000    44  ;       The transport mechanism is DECnet.  Netdriver is called directly
0000    45  ;       via the internal IRP mechanism.
0000    46  ;
0000    47  ; AUTHOR:
0000    48  ;
0000    49  ;       Len Kawell, 01-AUG-1979
0000    50  ;
0000    51  ; MODIFICATION HISTORY:
0000    52  ;
0000    53  ;       V03-014 JLV0390         Jake VanNoy             25-JUL-1984
0000    54  ;               Return ILLIOFUNC for FMS when PICSTRING is seen.
0000    55  ;
0000    56  ;       V03-013 LMP0275         L. Mark Pilant,         12-Jul-1984  12:42
0000    57  ;               Initialize the ACL info in the ORB to be a null descriptor
```

```
0000     58 ;                list rather than an empty queue.  This avoids the overhead
0000     59 ;                of locking and unlocking the ACL mutex, only to find out
0000     60 ;                that the ACL was empty.
0000     61 ;
0000     62 ;        V03-012 EMD0088         Ellen M. Dusseault      30-Apr-1984
0000     63 ;                Add DEV$M_NNM characteristic to DEVCHAR2 so that these
0000     64 ;                devices will have the "node$" prefix.
0000     65 ;
0000     66 ;        V03-011 LMP0221         L. Mark Pilant,         27-Mar-1984  11:53
0000     67 ;                Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
0000     68 ;                ORB$W_PROT.
0000     69 ;
0000     70 ;        V03-010 JLV0320         Jake VanNoy             18-DEC-1983
0000     71 ;                Remoe SS$_INCOMPAT from read fdt routine.  This error
0000     72 ;                is preventing set host from RSX and TOPS20.
0000     73 ;                Change write routine to send broadcast type message
0000     74 ;                if IO$M_BREAKTHRU is seen.  Remove RTT_BROADCAST routine
0000     75 ;                as it is obsolete.  Redo SET_MODE FDT to use case statement.
0000     76 ;                Clear io$m_extend bit in read routine. Remove CTRLC
0000     77 ;                and outband from SENSE_SPAWN.
0000     78 ;
0000     79 ;        V03-009 JLV0299         Jake VanNoy             30-JUL-1983
0000     80 ;                Add DEV$M_RTT to DPT_STORE's.
0000     81 ;
0000     82 ;        V03-008 JLV0252         Jake VanNoy             13-MAY-1983
0000     83 ;                Remove references to IO$M_ENABL_ALT and IO$M_DSABL_ALT.
0000     84 ;
0000     85 ;        V03-007 JLV0241         Jake VanNoy             20-APR-1983
0000     86 ;                Change ASSUME regarding TRM$_LASTITM.
0000     87 ;
0000     88 ;        V03-006 JLV0239         Jake VanNoy             29-MAR-1983
0000     89 ;                Add code to do new itemlist, remove V3.2 code to
0000     90 ;                handle read verify.
0000     91 ;
0000     92 ;        V03-005 JLV0227         Jake VanNoy             9-FEB-1983
0000     93 ;                Bug fix in error path of ALLOC_MESSAGE that caused
0000     94 ;                system crash. Another bug fix to the read fdt routine
0000     95 ;                that crashed system with large prompt size.
0000     96 ;
0000     97 ;        V03-004 JLV0215         Jake VanNoy             6-OCT-1982
0000     98 ;                Mods to SBL3007 to do parameter checking correctly.
0000     99 ;
0000    100 ;        V03-003 SBL3007         Steve Long      6-Aug-1982
0000    101 ;                Read verify support and permit IO$M_ENABL_ALT &
0000    102 ;                IO$M_DSABL_ALT to be processed in SETMODE
0000    103 ;
0000    104 ;        V03-002 DJD3007         Darrell Duffy   5-April-1982
0000    105 ;                Trap IO$M_ESCAPE and IO$M_EXTEND with reads to V2 systems.
0000    106 ;                Trap IO$M_ENABL_ALT IO$M_DSABL_ALT in SETMODE.
0000    107 ;
0000    108 ;        V03-001 DJD3006         Darrell Duffy   31-March-1982
0000    109 ;                Fix SENSEMODE TYPAHDCNT to return correct status.
0000    110 ;                Insert setting of mode bits for fixing spawn.
0000    111 ;
0000    112 ;        V02-016 DJD3005         Darrell Duffy   13-January-1982
0000    113 ;                Fix flushing of CTRL/Y to occur at deassign.
0000    114 ;                Use new cancel interface to distinguish cancel and deassign.
```

```
0000   115 ;
0000   116 ;       V02-015 DJD3004         Darrell Duffy   20-December-1981
0000   117 ;               Revert to use of attn ast processing for CTRL C and Y.
0000   118 ;               Remove privileges associated with declaring a ctrl/y ast.
0000   119 ;
0000   120 ;       V02-014 DJD3003         Darrell Duffy   24-November-1981
0000   121 ;               Add out-of-band ast support.  Fix bug in delivery
0000   122 ;               of hangup ast when the link has broken before it
0000   123 ;               was enabled.
0000   124 ;
0000   125 ;       V02-013 DJD3002         Darrell Duffy   12-November-1981
0000   126 ;               More of the same.
0000   127 ;
0000   128 ;       V02-012 DJD3001         Darrell Duffy   21-October-1981
0000   129 ;               Update for changes to terminal driver for V3.0
0000   130 ;
0000   131 ;       V02-011 DJD2004         Darrell Duffy   31-July-1981
0000   132 ;               Change broadcast interface to return failure on
0000   133 ;               terminal set for NOBROADCAST
0000   134 ;
0000   135 ;       V02-010 DJD2003         Darrell Duffy   2-May-1981
0000   136 ;               Fix double deallocate of rtt ucb.
0000   137 ;
0000   138 ;       V02-009 RLRLBCNT        Robert L. Rappaport     8-April-1981
0000   139 ;               Changes associated with IRP modifications to all BCNT
0000   140 ;               fields which have grown to longwords.  Also fix old bug
0000   141 ;               in RTT_WRITE which sometimes left garbage in R1.
0000   142 ;
0000   143 ;       V02-008 DJD2002         Darrell Duffy   8-Apr-1981
0000   144 ;               Fix race condition with broadcast messages after hangup.
0000   145 ;
0000   146 ;       V02-007 DJD2001         Darrell Duffy   5-Mar-1981
0000   147 ;               Change to call network driver directly to read and
0000   148 ;               write packets.
0000   149 ;
0000   150 ;       V02-006 LMK0006         Len Kawell      27-Feb-1981
0000   151 ;               Fix problem with immediate delivery of hangup AST when
0000   152 ;               AST is being cancelled.
0000   153 ;
0000   154 ;       1.05    LMK0005         Len Kawell      18-Mar-1980
0000   155 ;               Change broadcast to call EXE$ALONONPAGED.
0000   156 ;
0000   157 ;       1.04    LMK0004         Len Kawell      29-Feb-1980
0000   158 ;               Change adapter type in DPTAB to be NULL.
0000   159 ;
0000   160 ;       1.03    LMK0003         Len Kawell      25-Feb-1980
0000   161 ;               Change broadcast to not wait for completion to avoid
0000   162 ;               causing issuing process to indefinitely wait if delays
0000   163 ;               occur during remote delivery.
0000   164 ;
0000   165 ;       1.02    LMK0002         Len Kawell      21-Jan-1980
0000   166 ;               Add UCB$M_HANGUP flag so hangup is never lost.
0000   167 ;
0000   168 ;--
```

```
                        .SBTTL  External and local symbol definitions

                ;
                ; External symbols
                ;

                        $ACBDEF                         ; AST control block
                        $AQBDEF                         ; ACP queue block
                        $CANDEF                         ; Cancel interface codes
                        $CRBDEF                         ; Channel request block
                        $DCDEF                          ; Device classes and types
                        $DDBDEF                         ; Device data block
                        $DEVDEF                         ; Device characteristics
                        $DYNDEF                         ; Buffer type codes
                        $IDBDEF                         ; Interrupt data block
                        $IODEF                          ; I/O function codes
                        $IPLDEF                         ; Hardware IPL definitions
                        $IRPDEF                         ; I/O request packet
                        $JIBDEF                         ; Job Information block
                        $MSGDEF                         ; Mailbox message types
                        $ORBDEF                         ; OBJECT'S RIGHTS BLOCK OFFSETS
                        $PCBDEF                         ; Process control block
                        $PRDEF                          ; Processor registers
                        $PRVDEF                         ; Privilege bits
                        $PSLDEF                         ; Processor status longword
                        $RBFDEF                         ; Remote Device Buffer definitions
                        $RDPDEF                         ; Remote device packet
                        $REMDEF                         ; General constants
                        $SSDEF                          ; System status codes
                        $TRMDEF                         ; Item list definitions
                        $TTDEF                          ; Terminal definitions
                        $TT2DEF                         ; More definitions
                        $TTYDEF                         ; Terminal driver definitions
                        $UCBDEF                         ; Unit control block
                        $VCBDEF                         ; Volume control block
                        $VECDEF                         ; Interrupt vector block

                ;
                ; Local symbols
                ;

                ;
                ; Argument list (AP) offsets for device-dependent QIO parameters
                ;

00000000        P1      = 0                             ; First QIO parameter
00000004        P2      = 4                             ; Second QIO parameter
00000008        P3      = 8                             ; Third QIO parameter
0000000C        P4      = 12                            ; Fourth QIO parameter
00000010        P5      = 16                            ; Fifth QIO parameter
00000014        P6      = 20                            ; Sixth QIO parameter
```

```
                     0000   223
                     0000   224 ;
                     0000   225 ; Other constants
                     0000   226 ;
                     0000   227
00000008             0000   228 RTT$K_FIPL = 8                          ; IPL to synchronize
                     0000   229
                     0000   230 ;
                     0000   231 ; Definitions that follow the standard UCB fields
                     0000   232 ;
                     0000   233
                     0000   234         $RTTUCBEXT                     ; UCB Extensions
                     0000   235
000000DE             0000   236 UCB$W_RTT_READERR = UCB$W_CT_QCTPCNT    ; unused cterm UCB field
                     0000   237
                     0000   238 ;
                     0000   239 ; Redefinitions of the irp fields
                     0000   240 ;
                     0000   241
00000040             0000   242 IRP$W_RTT_COMPAT = IRP$Q_TT_STATE       ; Set for compatiblity error
                     0000   243
```

```
                                J 16
0000    245          .SBTTL  Standard tables
0000    246
0000    247 ;
0000    248 ; Driver prologue table
0000    249 ;
0000    250
0000    251          DPTAB   -                                       ; DPT-creation macro
0000    252                  END=RTT_END,-                           ; End of driver label
0000    253                  ADAPTER=NULL,-                          ; Adapter type
0000    254                  UCBSIZE=<UCB$K_RTT_LEN>,-               ; Length of UCB
0000    255                  NAME=RTTDRIVER                          ; Driver name
0038    256          DPT_STORE INIT                                  ; Start of load
0038    257                                                          ; initialization table
0038    258          DPT_STORE DDB,DDB$L_ACPD,L,<^A\REM\>            ; Default ACP name
003F    259          DPT_STORE DDB,DDB$L_ACPD+3,B,3                  ; ACP class
0043    260          DPT_STORE UCB,UCB$B_FIPL,B,RTT$K_FIPL           ; Device fork IPL
0047    261          DPT_STORE UCB,UCB$B_DIPL,B,RTT$K_FIPL           ; Device interrupt IPL
004B    262          DPT_STORE UCB,UCB$L_DEVCHAR,L,<-                ; Device characteristics
004B    263                  DEV$M_REC!-                             ;   record device
004B    264                  DEV$M_AVL!-                             ;   available
004B    265                  DEV$M_IDV!-                             ;   input device
004B    266                  DEV$M_ODV!-                             ;   output device
004B    267                  DEV$M_TRM!-                             ;   terminal device
004B    268                  DEV$M_CCL>                              ;   carriage control device
0052    269          DPT_STORE UCB,UCB$L_DEVCHAR2,L,<-               ; Device characteristics
0052    270                  DEV$M_RTT!-                             ; remote terminal UCB extension
0052    271                  DEV$M_NNM>                              ; prefix with "node$"
0059    272          DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_TERM         ; Terminal device
005D    273          DPT_STORE UCB,UCB$B_DEVTYPE,B,TT$_UNKNOWN       ; Unknown type
0061    274          DPT_STORE UCB,UCB$W_DEVBUFSIZ,aW,TTY$GW_DEFBUF  ; Default buffer size
0068    275          DPT_STORE UCB,UCB$L_DEVDEPEND,aL,TTY$GL_DEFCHAR ; Default characteristics
006F    276          DPT_STORE ORB,ORB$B_FLAGS,B,-                   ; Protection block flags
006F    277                          <ORB$M_PROT_16>                 ; SOGW protection word
0073    278          DPT_STORE ORB,ORB$W_PROT,aW,TTY$GW_PROT         ; Default allocation protection
007A    279          DPT_STORE ORB,ORB$L_OWNER,aL,TTY$GL_OWNUIC      ; Default owner UIC
0081    280
0081    281          DPT_STORE REINIT                                ; Start of reload
0081    282                                                          ; initialization table
0081    283          DPT_STORE DDB,DDB$L_DDT,D,RTT$DDT               ; Address of DDT
0086    284          DPT_STORE CRB,CRB$L_INTD+4,D,-                  ; Address of interrupt
0086    285                  RTT_INTERRUPT                           ; service routine
008B    286
008B    287          DPT_STORE END                                   ; End of initialization
0000    288                                                          ; tables
0000    289
0000    290 ;
0000    291 ; Driver dispatch table
0000    292 ;
0000    293
0000    294          DDTAB   -                                       ; DDT-creation macro
0000    295                  DEVNAM=RTT,-                            ; Name of device
0000    296                  FUNCTB=RTT_FUNCTABLE,-                  ; FDT address
0000    297                  UNSOLIC=RTT_UNSOLIC,-                   ; Unsolicited attention routine
0000    298                  CANCEL=RTT_CANCEL                       ; Cancel I/O routine
0038    299
0038    300 ;
0038    301 ; Function dispatch table
```

```
                     ;
                     ;
                     RTT_FUNCTABLE:                                  ; FDT for driver
                             FUNCTAB ,-                              ; Valid I/O functions
                                     <READVBLK,-                     ; Read virtual
                                     READLBLK,-                      ; Read logical
                                     READPBLK,-                      ; Read physical
                                     READPROMPT,-                    ; Read with prompt
                                     TTYREADALL,-                    ; Read passall
                                     TTYREADPALL,-                   ; Read with prompt passall
                                     WRITEVBLK,-                     ; Write virtual
                                     WRITELBLK,-                     ; Write logical
                                     WRITEPBLK,-                     ; Write physical
                                     SENSEMODE,-                     ; Sense device mode
                                     SENSECHAR,-                     ; Sense device characteristics
                                     SETMODE,-                       ; Set device mode
                                     SETCHAR>                        ; Set device characteristics
                             FUNCTAB ,-                              ; Buffered functions
                                     <READVBLK,-                     ; Read virtual
                                     READLBLK,-                      ; Read logical
                                     READPBLK,-                      ; Read physical
                                     READPROMPT,-                    ; Read with prompt
                                     TTYREADALL,-                    ; Read passall
                                     TTYREADPALL,-                   ; Read with prompt passall
                                     WRITEVBLK,-                     ; Write virtual
                                     WRITELBLK,-                     ; Write logical
                                     WRITEPBLK,-                     ; Write physical
                                     SENSEMODE,-                     ; Sense device mode
                                     SENSECHAR,-                     ; Sense device characteristics
                                     SETMODE,-                       ; Set device mode
                                     SETCHAR>                        ; Set device characteristics
                     FUNCTAB RTT_READ,-                              ; FDT read routine for
                                     <READVBLK,-                     ; read virtual,
                                     READLBLK,-                      ; read logical,
                                     READPBLK,-                      ; read physical,
                                     READPROMPT,-                    ; read with prompt
                                     TTYREADALL,-                    ; read passall,
                                     TTYREADPALL>                    ; and read with prompt passall
                     FUNCTAB RTT_WRITE,-                             ; FDT write routine for
                                     <WRITEVBLK,-                    ; write virtual,
                                     WRITELBLK,-                     ; write logical,
                                     WRITEPBLK>                      ; and write physical.
                     FUNCTAB RTT_SENSEMODE,-                         ; FDT sense mode routine
                                     <SENSECHAR,-                    ; for sense characteristics
                                     SENSEMODE>                      ; and sense mode.
                     FUNCTAB RTT_SETMODE,-                           ; FDT set mode routine
                                     <SETCHAR,-                      ; for set characteristics and
                                     SETMODE>                        ; set mode.
```

```
                     0078   351          .SBTTL  RTT_WRITE - Function Decision Routine for WRITE Functions
                     0078   352 ;++
                     0078   353 ; RTT_WRITE - Function Decision Routine for WRITE Functions
                     0078   354 ;
                     0078   355 ; Functional description:
                     0078   356 ;
                     0078   357 ;       This routine is called by the SYS$QIO service to dispatch a WRITE
                     0078   358 ;       I/O request.
                     0078   359 ;
                     0078   360 ;       The QIO parameters for terminal WRITES are:
                     0078   361 ;
                     0078   362 ;       P1 = address of the buffer
                     0078   363 ;       P2 = size of the buffer
                     0078   364 ;       P3 = (unused)
                     0078   365 ;       P4 = carriage control specifier
                     0078   366 ;
                     0078   367 ;       The buffer is validated for access, the process's quota checked and
                     0078   368 ;       decremented, the data and carriage control are copied to a message
                     0078   369 ;       block, the address of the message block is stored in the IRP,
                     0078   370 ;       and the IRP is queued to the ACP for delivery to the remote system.
                     0078   371 ;
                     0078   372 ; Inputs:
                     0078   373 ;
                     0078   374 ;       R0-R2 = scratch registers
                     0078   375 ;       R3 = address of the IRP (I/O request packet)
                     0078   376 ;       R4 = address of the PCB (process control block)
                     0078   377 ;       R5 = address of the UCB (unit control block)
                     0078   378 ;       R6 = address of the CCB (channel control block)
                     0078   379 ;       R7 = bit number of the I/O function code
                     0078   380 ;       R8 = address of the FDT table entry for this routine
                     0078   381 ;       R9-R11 = scratch registers
                     0078   382 ;       AP = address of the 1st function dependent QIO parameter
                     0078   383 ;
                     0078   384 ; Outputs:
                     0078   385 ;
                     0078   386 ;       IRP$L_SVAPTE(R3) = address of message buffer
                     0078   387 ;       IRP$W_BOFF(R3) = size of message buffer
                     0078   388 ;       IRP$W_BCNT(R3) = size of user buffer
                     0078   389 ;
                     0078   390 ;       The routine preserves all registers except R0-R2, and
                     0078   391 ;       R9-R11.
                     0078   392 ;
                     0078   393 ;--
                     0078   394 RTT_WRITE:                                  ; WRITE FDT routine
     56   6C    D0   0078   395          MOVL    P1(AP),R6                 ; Get user buffer virtual address
     50   56    D0   007B   396          MOVL    R6,R0                     ; Set up for write check call
 57    04 AC    3C   007E   397          MOVZWL  P2(AP),R7                 ; Get buffer size
     51   57    D0   0082   398          MOVL    R7,R1                     ; Set up for write check call
          06    13   0085   399          BEQL    10$                       ; Skip check if zero
00000000'GF    16   0087   400          JSB     G^EXE$WRITECHK            ; Check buffer access
                    008D   401                                            ; (no return means no access)
                    008D   402 ;
                    008D   403 ; Allocate the message buffer
                    008D   404 ;
                    008D   405 10$:
     51   20    C0   008D   406          ADDL    #RBF$T_TT_WDATA,R1        ; Add header to request size
        03EB    30   0090   407          BSBW    ALLOC_MESSAGE             ; Allocate the message buffer
```

```
                                0093    408 ;
                                0093    409 ; Copy the data and carriage control to the message
                                0093    410 ;
          18 A2    57    3C    0093    411         MOVZWL  R7,RBF$L_TT_BCNT(R2)    ; Set requested byte count
                   3C    BB    0097    412         PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
          54    20 A3    3C    0099    413         MOVZWL  IRP$W_FUNC(R3),R4       ; save function code and modifiers
          5A    0C AC    D0    009D    414         MOVL    P4(AP),R10              ; save carriage control
          09 54    09    E1    00A1    415         BBC     #IO$V_BREAKTHRU,R4,20$  ; Branch if not breakthru
                                00A5    416 ;
                                00A5    417 ; Format message so that it looks like the old broadcast message. Note
                                00A5    418 ; carriage control is cleared. This is a shortcoming
                                00A5    419 ; in this implementation, but this code will be obsolete shortly...
                                00A5    420 ;
          0E A2    01    AE    00A5    421         MNEGW   #1,RBF$W_OPCODE(R2)     ; Set function code for broadcast
                10 A2    B4    00A9    422         CLRW    RBF$W_MOD(R2)           ; No modifier bits here
                   5A    D4    00AC    423         CLRL    R10                     ; set no carriage control
                                00AE    424 20$:
       20 A2    66    57    28    00AE    425         MOVC3   R7,(R6),RBF$T_TT_WDATA(R2) ; Copy data
                51    53    D0    00B3    426         MOVL    R3,R1                   ; Save adr beyond data
                   3C    BA    00B6    427         POPR    #^M<R2,R3,R4,R5>        ; Restore the registers
          1C A2    5A    D0    00B8    428         MOVL    R10,RBF$L_TT_CARCON(R2) ; Copy carriage control
                                00BC    429 ;
                                00BC    430 ; Send the message to the remote device and exit QIO service
                                00BC    431 ;
             52    51    D0    00BC    432         MOVL    R1,R2                   ; Pointer beyond data in message
                40 A3    B4    00BF    433         CLRW    IRP$W_RTT_COMPAT(R3)    ; No compatibility error
                   06B6    31    00C2    434         BRW     RTT_NETMSGSENDX         ;
```

```
                     00C5   436          .SBTTL  RTT_READ - Function Decision Routine for READ Functions
                     00C5   437
                     00C5   438 ;++
                     00C5   439 ; RTT_READ - Function Decision Routine for READ Functions
                     00C5   440 ;
                     00C5   441 ; Functional description:
                     00C5   442 ;
                     00C5   443 ;       This routine is called by the SYS$QIO service to dispatch a READ
                     00C5   444 ;       I/O request.
                     00C5   445 ;
                     00C5   446 ;       The QIO parameters for terminal READS are:
                     00C5   447 ;
                     00C5   448 ;       P1 = address of the buffer
                     00C5   449 ;       P2 = size of the buffer
                     00C5   450 ;       P3 = number of seconds to wait for characters
                     00C5   451 ;       P4 = address of terminator class bitmask or 0 if standard
                     00C5   452 ;       P5 = address of prompt string for IO$_READPROMPT or IO$_TTYREADPALL
                     00C5   453 ;       P6 = size of prompt string for IO$_READPROMPT or IO$_TTYREADPALL
                     00C5   454 ;
                     00C5   455 ;       The buffer is validated for access, the process's quota checked and
                     00C5   456 ;       decremented, the timeout, terminator mask, and prompt are copied to a
                     00C5   457 ;       message block, the address of the message block is stored in the IRP,
                     00C5   458 ;       and the IRP is queued to the ACP for delivery to the remote system.
                     00C5   459 ;
                     00C5   460 ; Inputs:
                     00C5   461 ;
                     00C5   462 ;       R0-R2 = scratch registers
                     00C5   463 ;       R3 = address of the IRP (I/O request packet)
                     00C5   464 ;       R4 = address of the PCB (process control block)
                     00C5   465 ;       R5 = address of the UCB (unit control block)
                     00C5   466 ;       R6 = address of the CCB (channel control block)
                     00C5   467 ;       R7 = bit number of the I/O function code
                     00C5   468 ;       R8 = address of the FDT table entry for this routine
                     00C5   469 ;       R9-R11 = scratch registers
                     00C5   470 ;       AP = address of the 1st function dependent QIO parameter
                     00C5   471 ;
                     00C5   472 ; Outputs:
                     00C5   473 ;
                     00C5   474 ;       IRP$L_SVAPTE(R3) = address of message buffer
                     00C5   475 ;       IRP$W_BOFF(R3) = size of message buffer
                     00C5   476 ;       IRP$L_MEDIA(R3) = address of user buffer
                     00C5   477 ;       IRP$W_BCNT(R3) = size of user buffer
                     00C5   478 ;
                     00C5   479 ;       The routine preserves all registers except R0-R2, and
                     00C5   480 ;       R9-R11.
                     00C5   481 ;
                     00C5   482 ;--
                     00C5   483
                     00C5   484 ; Local storage offsets on stack:
                     00C5   485
00000000             00C5   486 bufaddr = 0
00000004             00C5   487 bufsize = 4
00000008             00C5   488 prmaddr = 8
0000000C             00C5   489 prmsize = 12
00000010             00C5   490 trmaddr = 16
00000014             00C5   491 trmsize = 20
00000018             00C5   492 iniaddr = 24
```

```
                00000001C  00C5    493 inisize = 28
                00000020   00C5    494 timeout = 32
                00000024   00C5    495 inioffset = 36
                           00C5    496
                00000028   00C5    497 read_local = 40
                           00C5    498
                           00C5    499 RTT_READ:                           ; READ FDT routine
                           00C5    500 ;
                           00C5    501 ; Set up stack locals
                           00C5    502 ;
          5E    28   C2    00C5    503         SUBL2   #READ_LOCAL,SP      ; Allocate local storage
          58    5E   D0    00C8    504         MOVL    SP,R8               ; Save pointer
                68   7C    00CB    505         CLRQ    (R8)                ; clear buf ***
                08 A8   7C  00CD    506         CLRQ    8(R8)               ; clear prm ...
                10 A8   7C  00D0    507         CLRQ    16(R8)              ; clear trm ...
                18 A8   7C  00D3    508         CLRQ    24(R8)              ; clear ini ...
                20 A8   7C  00D6    509         CLRQ    32(R8)              ; clear other ...
                           00D9    510 ;
                           00D9    511 ; Check access to user's buffer
                           00D9    512 ;
          50    6C   D0    00D9    513         MOVL    P1(AP),R0           ; Get user buffer virtual address
       38 A3    50   D0    00DC    514         MOVL    R0,IRP$L_MEDIA(R3)  ; Save address in packet
       51    04 AC   3C    00E0    515         MOVZWL  P2(AP),R1           ; Get buffer size
                09   13    00E4    516         BEQL    10$                 ; Skip check if zero
             68    50   7D    00E6    517         MOVQ    R0,BUFADDR(R8)      ; Set up for read check call
00000000'GF    16   00E9    518         JSB     G^EXE$READCHK               ; Check buffer access
                           00EF    519                                     ; (no return means no access)
                           00EF    520 ;
                           00EF    521 ; Check for extended itemlist read
                           00EF    522 ;
                           00EF    523 10$:
                0F   E5    00EF    524         BBCC    #IO$V_EXTEND,-      ; If this is not item list
          06 20 A3         00F1    525                 IRP$W_FUNC(R3),15$  ; then continue
                00A9   30  00F4    526         BSBW    RT_READ_ITMLST      ; process item list
                0059   31  00F7    527         BRW     200$                ; continue
                           00FA    528 ;
                           00FA    529 ; Get prompt, if specified
                           00FA    530 ;
                           00FA    531 15$:
          37    57   91    00FA    532         CMPB    R7,#IO$_READPROMPT  ; Read prompt?
                05   13    00FD    533         BEQL    20$                 ; Branch if yes
          3B    57   91    00FF    534         CMPB    R7,#IO$_TTYREADPALL ; Read prompt?
                14   12    0102    535         BNEQ    50$                 ; Branch if not
       51    14 AC   3C    0104    536 20$:    MOVZWL  P6(AP),R1           ; Get size of prompt
                0E   13    0108    537         BEQL    50$                 ; If eql then make this normal read
       50    10 AC   D0    010A    538         MOVL    P5(AP),R0           ; Get prompt buffer address
                           010E    539 ;
                           010E    540 ; Check access to prompt string
                           010E    541 ;
       08 A8    50   7D    010E    542         MOVQ    R0,PRMADDR(R8)      ; Save address and size
00000000'GF    16   0112    543         JSB     G^EXE$WRITECHK              ; Check prompt access
                           0118    544 ;
                           0118    545 ; Get terminator bitmask and check access
                           0118    546 ;
                           0118    547 50$:
                52   D4    0118    548         CLRL    R2                  ; Assume no terminator specified
       51    0C AC   D0    011A    549         MOVL    P4(AP),R1           ; Get address of terminator desc
```

```
                 2A   13   011E   550          BEQL    65$                     ; If eql none specified
              50 0C   3C   0120   551          MOVZWL  #SS$_ACCVIO,R0          ; Assume no access
                      0123   552          IFNORD  #8,(R1),63$             ; Descriptor accessible?
              52 61   3C   0129   553          MOVZWL  (R1),R2                 ; Get bitmask size
                 08   12   012C   554          BNEQ    60$                     ; If neq long format
              52 04   D0   012E   555          MOVL    #4,R2                   ; Size of short format
              51 04   C0   0131   556          ADDL    #4,R1                   ; Set address of bitmask
                 14   11   0134   557          BRB     65$                     ;
                      0136   558 60$:
           51   04 A1   D0   0136   559          MOVL    4(R1),R1                ; Get address of long format bitmask
                      013A   560          IFNORD  R2,(R1),63$             ; Bitmask accessible?
              20 52   B1   0140   561          CMPW    R2,#32                  ; Bitmask greater than allowed size?
                 05   1B   0143   562          BLEQU   65$                     ; If gtru yes
              50 14   3C   0145   563          MOVZWL  #SS$_BADPARAM,R0        ; bad parameter
                 50   11   0148   564 63$:     BRB     READ_ERROR
                      014A   565 65$:
        10 A8 51   7D   014A   566          MOVQ    R1,TRMADDR(R8)          ; terminator address and size
     20 A8   08 AC   D0   014E   567          MOVL    P3(AP),TIMEOUT(R8)      ; Set timeout value
                      0153   568 200$:
                      0153   569 ;
                      0153   570 ; Commom code again, Allocate the message buffer
                      0153   571 ;
           5B   04 A8   D0   0153   572          MOVL    BUFSIZE(R8),R11         ; Set size of read
        32 A3   5B   B0   0157   573          MOVW    R11,IRP$W_BCNT(R3)      ; Reset read buffer size
                      015B   574                                          ; (modified by EXE$WRITECHK)
                      015B   575
              51   23   D0   015B   576          MOVL    #RBF$T_TT_TERM+3,R1     ; Set header + overhead size
           51   0C A8   C0   015E   577          ADDL    PRMSIZE(R8),R1          ; Prompt size
           51   14 A8   C0   0162   578          ADDL    TRMSIZE(R8),R1          ; terminator size
                0315   30   0166   579          BSBW    ALLOC_MESSAGE           ; Allocate the message buffer
                      0169   580 ;
                      0169   581 ; Copy the timeout, terminator bitmask, and prompt string to the message
                      0169   582 ;
        18 A2   5B   D0   0169   583          MOVL    R11,RBF$L_TT_BCNT(R2)   ; Set requested byte count
              20 A8   D0   016D   584          MOVL    TIMEOUT(R8),-
                 1C A2   0170   585                  RBF$L_TT_TIMOUT(R2)     ; Set timeout value
                 3C   BB   0172   586          PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
                      0174   587
           50   10 A8   7D   0174   588          MOVQ    TRMADDR(R8),R0          ; Set terminator addr and size
        20 A2   51   90   0178   589          MOVB    R1,RBF$T_TT_TERM(R2)    ; Set terminator bitmask size
     21 A2   60   51   28   017C   590          MOVC    R1,(R0),RBF$T_TT_TERM+1(R2) ; Copy terminator bitmask
                      0181   591
           50   08 A8   7D   0181   592          MOVQ    PRMADDR(R8),R0          ; Set prompt addr and size
              83   51   B0   0185   593          MOVW    R1,(R3)+                ; Set size of prompt
        63    60   51   28   0188   594          MOVC    R1,(R0),(R3)            ; Copy prompt string
                      018C   595
              51   53   D0   018C   596          MOVL    R3,R1                   ; Save adr beyond data
                 3C   BA   018F   597          POPR    #^M<R2,R3,R4,R5>        ; Restore registers
                      0191   598 ;
                      0191   599 ; Send the message the remote device and exit the QIO service
                      0191   600 ;
              52   51   D0   0191   601          MOVL    R1,R2                   ; Set address beyond data
                40 A3   B4   0194   602          CLRW    IRP$W_RTT_COMPAT(R3)    ; No compatiblity error
                 05E1   31   0197   603          BRW     RTT_NETMSGSENDX         ;
                      019A   604 ;
                      019A   605 ; Error in processing
                      019A   606 ;
```

```
                     019A   607 READ_ERROR:                          ; READ FDT error
00000000'GF    17    019A   608          JMP     G^EXE$ABORTIO       ; Abort the I/O request
```

```
                                                F 1
                                01A0   610          .SBTTL  RT_READ_ITMLST - FDT routine for read with item list
                                01A0   611 ;++
                                01A0   612 ;
                                01A0   613 ;
                                01A0   614 ;
                                01A0   615 ; *** a clean up pass is needed to here to verify that the paranoia
                                01A0   616 ;     checks made by TTDRIVER and this driver are the same.
                                01A0   617 ;
                                01A0   618 ;--
                                01A0   619
                                01A0   620 RT_READ_ITMLST:
                                01A0   621
                                01A0   622          ;
                                01A0   623          ; Set up probe of itemlist with P3 as access mode
                                01A0   624          ;
              56    53    D0    01A0   625          MOVL    R3,R6                   ; Save IRP
50  08 AC     02    00    EF    01A3   626          EXTZV   #0,#2,P3(AP),R0         ; fetch low 2 bits of parameter
        00000000'GF   16    01A9   627          JSB     G^EXE$MAXACMODE         ; maximize with mode of caller
              53    50    D0    01AF   628          MOVL    R0,R3                   ; Set input to probe routine
                                01B2   629
              50   10 AC    D0    01B2   630          MOVL    P5(AP),R0               ; Address of itemlist
              51   14 AC    D0    01B6   631          MOVL    P6(AP),R1               ; size of item list
                    05    13    01BA   632          BEQL    10$                     ; can't be zero?
              5A    50    7D    01BC   633          MOVQ    R0,R10                  ; save both
                    08    11    01BF   634          BRB     30$                     ; ok, continue
              50    14    3C    01C1   635 10$:     MOVZWL  #SS$_BADPARAM,R0        ; status
              53    56    D0    01C4   636 20$:     MOVL    R6,R3                   ; Restore IRP
                    D1    11    01C7   637          BRB     READ_ERROR              ; abort
                                01C9   638
        00000000'GF   16    01C9   639 30$:     JSB     G^EXE$PROBER            ; Can it be read?
              F2    50    E9    01CF   640          BLBC    R0,20$                  ; branch if not
              50    5B    D0    01D2   641          MOVL    R11,R0                  ; size
                                01D5   642          ;
                                01D5   643          ; Verify that size is multiple of 12
                                01D5   644          ;
              53    56    D0    01D5   645          MOVL    R6,R3                   ; Restore IRP
                    51    D4    01D8   646          CLRL    R1                      ; quadword r0/r1
50    5B    50    0C    7B    01DA   647          EDIV    #12,R0,R11,R0           ; divide
                    50    D5    01DF   648          TSTL    R0                      ; must be zero remainder
                    DE    12    01E1   649          BNEQ    10$                     ; error
                                01E3   650          ;
                                01E3   651          ; Now loop and conquer item list, item by item
                                01E3   652          ;
                                01E3   653 40$:
              51    8A    3C    01E3   654          MOVZWL  (R10)+,R1               ; Length
              52    8A    3C    01E6   655          MOVZWL  (R10)+,R2               ; item code
              50    8A    D0    01E9   656          MOVL    (R10)+,R0               ; address or immediate value
                    8A    D5    01EC   657          TSTL    (R10)+                  ; Must be zero field
                    D1    12    01EE   658          BNEQ    10$                     ; error if not
                                01F0   659
                                01F0   660          CASE    R2,-                    ; case on message type
                                01F0   661                  <100$,-                 ; TRM$_MODIFIERS        (0)
                                01F0   662                  200$,-                  ; TRM$_EDITMODE         (1)
                                01F0   663                  300$,-                  ; TRM$_TIMEOUT          (2)
                                01F0   664                  400$,-                  ; TRM$_TERM             (3)
                                01F0   665                  500$,-                  ; TRM$_PROMPT           (4)
                                01F0   666                  600$,-                  ; TRM$_INISTRING        (5)
```

```
                        01F0   667                     700$,-                          ; TRM$_PICSTRING        (6)
                        01F0   668                     800$,-                          ; TRM$_FILLCHR          (7)
                        01F0   669                     900$,-                          ; TRM$_INIOFFSET        (8)
                        01F0   670                     1000$-                          ; TRM$_ALTECHSTR        (9)
                        01F0   671                     >,-                             ; TRM$_LASTITM          (10)
                        01F0   672                     TYPE = W
                        0208   673
                        0208   674             ASSUME  TRM$_LASTITM EQ 10              ; Break assembly if not right
                B7  11  0208   675             BRB     10$
                        020A   676
                        020A   677 100$:                        ; TRM$_MODIFIERS
                        020A   678
          50  8000 8F  AA  020A   679             BICW    #IO$M_EXTEND,R0                 ; clear extend bit
        20 A3   50  A8  020F   680             BISW    R0,IRP$W_FUNC(R3)               ; Set read flags
                5A  11  0213   681             BRB     2000$                           ; Loop
                        0215   682
                        0215   683 200$:                        ; TRM$_EDITMODE
                58  11  0215   684             BRB     2000$                           ; ignore
                        0217   685
                        0217   686 300$:                        ; TRM$_TIMEOUT
                        0217   687
        20 A8   50  D0  0217   688             MOVL    R0,TIMEOUT(R8)                  ; Set timeout
    20 A3   0080 8F  A8  021B   689             BISW    #IO$M_TIMED,IRP$W_FUNC(R3) ; set read timed bit
                4C  11  0221   690             BRB     2000$                           ; loop
                        0223   691
                        0223   692 400$:                        ; TRM$_TERM
                51  D5  0223   693             TSTL    R1                              ; test length
                09  12  0225   694             BNEQ    410$                            ; If neq long format
            51  04  D0  0227   695             MOVL    #4,R1                           ; Size of short format
          50  F8  AA  9E  022A   696             MOVAB   -8(R10),R0                      ; Address of immediate data *** hack
                13  11  022E   697             BRB     430$                            ; skip
                        0230   698 410$:
                        0230   699             IFNORD  R1,(R0),420$                    ; Bitmask accessible?
            20  51  B1  0236   700             CMPW    R1,#32                          ; Bitmask greater than allowed size?
                08  18  0239   701             BLEQU   430$                            ; If less than or equal, no
                84  11  023B   702             BRB     10$                             ; bad param *** other status?
            50  0C  3C  023D   703 420$:       MOVZWL  #SS$_ACCVIO,R0                  ; access violation
              FF57  31  0240   704             BRW     READ_ERROR                      ; branch to read error
                        0243   705 430$:
        10 A8   50  7D  0243   706             MOVQ    R0,TRMADDR(R8)                  ; save address and size of terminators
                26  11  0247   707             BRB     2000$                           ; continue
                        0249   708
                        0249   709 500$:                        ; TRM$_PROMPT
        08 A8   50  7D  0249   710             MOVQ    R0,PRMADDR(R8)                  ; save address and length
                37  F0  024D   711             INSV    #IO$_READPROMPT,-
            06  00       024F   712                     #IRP$V_FCODE,#IRP$S_FCODE,-
              20 A3     0251   713                     IRP$W_FUNC(R3)                  ; Set Read with prompt
                0C  11  0253   714             BRB     650$                            ; continue
                        0255   715
                        0255   716 700$:                        ; TRM$_PICSTRING
    50  00F4 8F  3C  0255   717             MOVZWL  #SS$_ILLIOFUNC,R0               ; for FMS...
              FF3D  31  025A   718             BRW     READ_ERROR
                        025D   719
                        025D   720 1000$:                       ; TRM$_ALTECOSTR
                        025D   721 600$:                        ; TRM$_INISTRING
        18 A8   50  7D  025D   722             MOVQ    R0,INIADDR(R8)                  ; save address and length
                51  D5  0261   723 650$:       TSTL    R1                              ; no need to check if zero
```

```
                 0A   13  0263   724          BEQL    2000$                           ; Skip parameter
                 0F   10  0265   725          BSBB    CHK_READERR                     ; check for read error
                 06   11  0267   726          BRB     2000$                           ; continue
                          0269   727
                          0269   728 800$:                    ; TRM$_FILLCHR
                          0269   729 900$:                    ; TRM$_INIOFFSET
                 50   B5  0269   730          TSTW    R0                              ; test to see if present
                 02   13  026B   731          BEQL    2000$                           ; branch if not
                 07   10  026D   732          BSBB    CHK_READERR                     ; check for read error
                          026F   733
                          026F   734 2000$:
              01 5B   F5  026F   735          SOBGTR  R11,2010$                       ; loop
                      05  0272   736          RSB
                          0273   737
                 FF6D 31  0273   738 2010$:  BRW     40$                             ;
                          0276   739
                          0276   740 CHK_READERR:
                          0276   741
           50 00DE C5   3C  0276   742          MOVZWL  UCB$W_RTT_READERR(R5),R0 ; set status
                 01   B0  027B   743          MOVW    #SS$_NORMAL,-
              00DE C5         027D   744                  UCB$W_RTT_READERR(R5)    ; set success if this happens again
                01 50  E9  0280   745          BLBC    R0,10$                          ; branch if error
                      05  0283   746          RSB                                     ; continue without error
                 FF13 31  0284   747 10$:    BRW     READ_ERROR                      ; abort
                          0287   748
```

```
                                .SBTTL  RTT_SETMODE, Function Decision Routine for SETMODE/SETCHAR
                        ;++
                        ; RTT_SETMODE, Function Decision Routine for SETMODE/SETCHAR Functions
                        ;
                        ; Functional description:
                        ;
                        ;       This routine is called by the SYS$QIO service to dispatch a SETMODE
                        ;       or SETCHAR I/O request.
                        ;
                        ;       The QIO parameters for terminal SETMODE or SETCHAR are:
                        ;
                        ;               P1 = address of 8 byte characteristics buffer
                        ;               P2 = 0, 8 or 12
                        ;               P3 = speed specifier
                        ;               P4 = fill specifier
                        ;               P5 = parity flags
                        ;
                        ;       IO$V_CTRLYAST -
                        ;               P1 = AST routine address or zero to cancel
                        ;
                        ;       IO$V_CTRLCAST -
                        ;               P1 = AST routine address or zero to cancel
                        ;
                        ;       IO$V_HANGUP -
                        ;               NONE
                        ;
                        ;       The buffer (if any) is validated for access, the process's quota
                        ;       checked and decremented, a message block is allocated, the parameters
                        ;       (if any) are stored in the  message block, the address of the message
                        ;       block is stored in the IRP, and the IRP is queued to the ACP for
                        ;       delivery to the remote system.
                        ;
                        ;       If an AST is to be enabled, an AST control block is allocated locally
                        ;       hung off the UCB for later delivery upon receipt of a corresponding
                        ;       attention message from the remote system.
                        ;
                        ; Inputs:
                        ;
                        ;       R0-R2 = scratch registers
                        ;       R3 = address of the IRP (I/O request packet)
                        ;       R4 = address of the PCB (process control block)
                        ;       R5 = address of the UCB (unit control block)
                        ;       R6 = address of the CCB (channel control block)
                        ;       R7 = bit number of the I/O function code
                        ;       R8 = address of the FDT table entry for this routine
                        ;       R9-R11 = scratch registers
                        ;       AP = address of the 1st function dependent QIO parameter
                        ;
                        ; Outputs:
                        ;
                        ;       IRP$L_SVAPTE(R3) = address of message buffer
                        ;       IRP$W_BOFF(R3) = size of message buffer
                        ;
                        ;       The routine preserves all registers except R0-R2, R7, and R9-R11
                        ;--
0287                    RTT_SETMODE:                            ; SETMODE/SETCHAR FDT routine
```

```
                     40 A3   B4  0287   807          CLRW    IRP$W_RTT_COMPAT(R3)    ; No compatibility error
                  50 20 A3   3C  028A   808          MOVZWL  IRP$W_FUNC(R3),R0       ; Fetch function code and modifers
              51  50 09 06   EA  028E   809          FFS     #IO$V_MAINT,#9,R0,R1    ; Find first set modifier
                     33      13  0293   810          BEQL    SET_CHAR                ; if none then simple set mode.
                             0295   811
                  50 0380 8F B3  0295   812          BITW    #<IO$M_CTRLCAST!-
                             029A   813                  IO$M_CTRLYAST!-
                             029A   814                  IO$M_HANGUP>,R0         ; Always legal functions
                        0E   12  029A   815          BNEQ    30$                     ; branch if any of these
                             029C   816
                     00D5 C5   95  029C   817          TSTB    UCB$B_RTT_PROECO(R5)    ; Previous version
                        08   12  02A0   818          BNEQ    30$                     ; Nope
                  50 069C 8F   3C  02A2   819          MOVZWL  #SS$_INCOMPAT+3, R0     ; Abort maintenance, outband, etc.
                        010F   31  02A7   820          BRW     ABORT                   ; with an error not success
                             02AA   821 30$:
                             02AA   822          CASE    R1,TYPE=B,LIMIT=#IO$V_MAINT,<-
                             02AA   823                  SET_MAINT,-             ; IO$M_MAINT
                             02AA   824                  SET_CTRLY,-             ; IO$M_CTRLYAST
                             02AA   825                  SET_CTRLC,-             ; IO$M_CTRLCAST
                             02AA   826                  SET_HANGUP,-            ; IO$M_HANGUP
                             02AA   827                  SET_OUTBAND,-           ; IO$M_OUTBAND
                             02AA   828                  SET_CONNECT,-           ; IO$M_CONNECT
                             02AA   829                  SET_DISCONNECT,-        ; IO$M_DISCONNECT
                             02AA   830                  SET_PID,-               ; IO$M_SETPID
                             02AA   831                  SET_BRDCST>             ; IO$M_BRDCST
                             02C0   832 ;
                             02C0   833 ; invalid characteristic if CASE falls though
                             02C0   834 ;
                  50 00F4 8F   3C  02C0   835          MOVZWL  #SS$_ILLIOFUNC, R0      ; Return as illegal operation
                        00F1   31  02C5   836          BRW     ABORT                   ; with an error not success
                             02C8   837
                             02C8   838 SET_CHAR:
                        00FD   30  02C8   839          BSBW    GET_PARAMS              ; validate and fetch parameters
              5B     48 A5   D0  02CB   840          MOVL    UCB$L_DEVDEPND2(R5),R11 ; Extended word is defaulted
                 59     81   7D  02CF   841          MOVQ    (R1)+,R9                ; Get characteristics
                 0C     52   D1  02D2   842          CMPL    R2, #12                 ; Do we get another longword?
                        03   19  02D5   843          BLSS    20$                     ; Nope
                 5B     81   D0  02D7   844          MOVL    (R1)+, R11              ; Obtain the third longword
              40 A5     59   7D  02DA   845 20$:     MOVQ    R9,UCB$B_DEVCLASS(R5)   ; Set local copy of characteristics
              48 A5     5B   D0  02DE   846          MOVL    R11,UCB$L_DEVDEPND2(R5) ; And extended longword
                             02E2   847
                     00D5 C5   95  02E2   848          TSTB    UCB$B_RTT_PROECO(R5)    ; If old version
                        10   12  02E6   849          BNEQ    30$                     ; Nope
                00F00000 8F   D3  02E8   850          BITL    # <<<1@24>-1>-<<1@TT$V_HALFDUP>-1>>,-
                     44 A5        02EE   851                  UCB$L_DEVDEPEND(R5)     ; If extra bits set, then
                        06   13  02F0   852          BEQL    30$                     ; return incompat error
                  0699 8F   B0  02F2   853          MOVW    #SS$_INCOMPAT,-         ; but carry on with function
                     40 A3        02F6   854                  IRP$W_RTT_COMPAT(R3)    ;
                             02F8   855 30$:
                        004F   31  02F8   856          BRW     SET_MESSAGE             ; send message
                             02FB   857          ;
                             02FB   858          ; The following types of modifiers are not allowed on remote terminals
                             02FB   859          ;
                             02FB   860 SET_MAINT:
                             02FB   861 SET_CONNECT:
                             02FB   862 SET_DISCONNECT:
                             02FB   863
```

```
                                                K 1
50   0334 8F   3C  02FB   864          MOVZWL  #SS$_DEVREQERR, R0       ; Return as device request error
        00B6   31  0300   865          BRW     ABORT                    ; with an error not success
                   0303   866
                   0303   867 SET_BRDCST:
        00C2   30  0303   868          BSBW    GET_PARAMS               ; Get parameters
00A8 C5   61   7D  0306   869          MOVQ    (R1),UCB$Q_TL_BRKTHRU(R5); Set bits
          06   11  030B   870          BRB     SET_NOP                  ; Set done
                   030D   871
                   030D   872 SET_PID:
      60 A4    D0  030D   873          MOVL    PCB$L_PID(R4),-
    00A4 C5        0310   874                  UCB$L_TL_CTLPID(R5)      ; Set controlling PID
                   0313   875 SET_NOP:
        00A9   31  0313   876          BRW     FDT_FINISHIOC_OK         ; Complete I/O
                   0316   877
                   0316   878 SET_CTRLY:
57   0090 C5   DE  0316   879          MOVAL   UCB$L_RTT_CTRLY(R5),R7   ; Get address of CNTRL/Y AST list
 00000000'GF   16  031B   880          JSB     G^COM$SETATTNAST         ; Enable an attention AST
          03   E1  0321   881          BBC     #UCB$V_TT_HANGUP,-
    21 68 A5        0323   882                  UCB$W_DEVSTS(R5),CTRL_CY ; Branch if no hangup
     50   67   D0  0326   883          MOVL    (R7),R0                  ; Get address of AST block
          1C   13  0329   884          BEQL    CTRL_CY                  ; If eql, no AST to deliver
     54   57   D0  032B   885          MOVL    R7,R4                    ; Set address of AST listhead
1C A0   02CC 8F   3C  032E   886          MOVZWL  #SS$_HANGUP,ACB$L_KAST+4(R0) ; Set AST parameter to hangup
 00000000'GF   16  0334   887          JSB     G^COM$DELATTNAST         ; Deliver the AST immediately
          D7   11  033A   888          BRB     SET_NOP                  ; finish I/O
                   033C   889
                   033C   890 SET_CTRLC:
57   0094 C5   DE  033C   891          MOVAL   UCB$L_RTT_CTRLC(R5),R7   ; set CNTRL/C AST enable
 00000000'GF   16  0341   892          JSB     G^COM$SETATTNAST         ; Enable an attention AST
                   0347   893
                   0347   894 CTRL_CY:
     59   6C   D0  0347   895          MOVL    P1(AP),R9                ; Get address of AST routine
                   034A   896                                           ; fall htrough to send message
                   034A   897 ;
                   034A   898 ; Create SET message and send to remote device
                   034A   899 ;
                   034A   900 SET_HANGUP:
                   034A   901 SET_MESSAGE:                             ; Create and queue SET message
     51   30   D0  034A   902          MOVL    #RBF$L_TT_CHAR2+4,R1     ; Set size of message buffer
        012E   30  034D   903          BSBW    ALLOC_MESSAGE            ; Allocate a message buffer
  18 A2   59   7D  0350   904          MOVQ    R9,RBF$Q_TT_CHAR(R2)     ; Set characteristics or AST parameter
  2C A2   5B   D0  0354   905          MOVL    R11, RBF$L_TT_CHAR2(R2)  ; And the next longword
20 A2   08 AC   D0  0358   906          MOVL    P3(AP),RBF$L_TT_SPEED(R2) ; Set speed
24 A2   0C AC   D0  035D   907          MOVL    P4(AP),RBF$L_TT_FILL(R2) ; Set fills
28 A2   10 AC   D0  0362   908          MOVL    P5(AP),RBF$L_TT_PARITY(R2) ; Set parity
 01   00D5 C5   91  0367   909          CMPB    UCB$B_RTT_PROECO(R5), - ; How long should the message be?
                   036C   910                  #REM$C_CURECO            ; Long or short
          06   12  036C   911          BNEQ    10$                      ; Shorter message
  52   30 A2   9E  036E   912          MOVAB   RBF$L_TT_CHAR2+4(R2),R2  ; Address of longer message
          04   11  0372   913          BRB     20$
  52   2C A2   9E  0374   914 10$:     MOVAB   RBF$L_TT_PARITY+4(R2),R2; Set address beyond data
        0400   31  0378   915 20$:     BRW     RTT_NETMSGSENDX          ; Send message to remote and exit service
                   037B   916
                   037B   917 ;
                   037B   918 ; Process a setmode for an outofband ast
                   037B   919 ;
                   037B   920
```

```
                     037B   921 SET_OUTBAND:
0C 20 A3   0B   E0   037B   922          BBS     #IO$V_INCLUDE, -          ; Include list?
                     0380   923                  IRP$W_FUNC(R3), 10$       ;
      009C C5   9E   0380   924          MOVAB   UCB$L_RTT_BANDEXCL(R5),-  ; Address of exclude ast list
             57      0384   925                  R7                        ;
      0098 C5   9E   0385   926          MOVAB   UCB$L_RTT_BANDEXMSK(R5),- ; Address of the exclude mask
             52      0389   927                  R2                        ;
                0A   11   038A   928          BRB     20$
                     038C   929
      00C4 C5   9E   038C   930 10$:     MOVAB   UCB$L_RTT_BANDINCL(R5),-  ; Address of include ast list
             57      0390   931                  R7                        ;
      00C8 C5   9E   0391   932          MOVAB   UCB$L_RTT_BANDINMSK(R5),- ; Address of the include mask
             52      0395   933                  R2                        ;
                     0396   934 20$:
00000000'GF   16   0396   935          JSB     G^COM$SETCTRLAST          ; Enable the asts
          22   D0   039C   936          MOVL    #RBF$B_TT_OUTBAND+1+4+1+4,-;
          51        039E   937                  R1                        ; Set size of message
        00DC   30   039F   938          BSBW    ALLOC_MESSAGE             ; Allocate a message
       18 A2   9E   03A2   939          MOVAB   RBF$B_TT_OUTBAND(R2),-    ; Address of data in message
          52        03A5   940                  R2                        ;
       82   04   90   03A6   941          MOVB    #4, (R2)+                 ; Count for include mask
      00C8 C5   D0   03A9   942          MOVL    UCB$L_RTT_BANDINMSK(R5),-; Include mask
          82        03AD   943                  (R2)+                     ;
       82   04   90   03AE   944          MOVB    #4, (R2)+                 ; Count for exclude mask
      0098 C5   D0   03B1   945          MOVL    UCB$L_RTT_BANDEXMSK(R5),- ; Now the exclude mask
          82        03B5   946                  (R2)+                     ;
        03C2   31   03B6   947          BRW     RTT_NETMSGSENDX           ; Send the message
```

```
                                                M 1
                     03B9   949          .SBTTL  ABORT, Transfer to EXE$ABORTIO
                     03B9   950
                     03B9   951 ;
                     03B9   952 ; Error processing - abort I/O request
                     03B9   953 ;
                     03B9   954 ABORT:                                       ;
00000C00'GF    17    03B9   955          JMP     G^EXE$ABORTIO               ;
                     03BF   956
                     03BF   957 ;
                     03BF   958 ; Finish I/O, clear R1
                     03BF   959 ;
                     03BF   960 FDT_FINISHIOC_OK:
     50    01    3C  03BF   961          MOVZWL  #SS$_NORMAL,R0              ; Set status OK
                     03C2   962 FDT_FINISHIOC:
00000000'GF    17    03C2   963          JMP     G^EXE$FINISHIOC             ; Complete I/O request
                     03C8   964
                     03C8   965          .SBTTL  GET_PARAMS - Get set mode parameters
```

```
                        03C8   967 ;++
                        03C8   968 ; GET_PARAMS
                        03C8   969 ;
                        03C8   970 ; inputs
                        03C8   971 ;       AP -> qio argument list
                        03C8   972 ;
                        03C8   973 ; outputs
                        03C8   974 ;       r1 = address of parameters
                        03C8   975 ;       r2 = 8 or 12 for size of characteristics buffer
                        03C8   976 ;
                        03C8   977 ; ABORT if P2(ap) is not 0, 8, 12.
                        03C8   978 ; Return ss$_incompat if not current system and size is 12.
                        03C8   979 ;--
                        03C8   980
                        03C8   981 GET_PARAMS:
                        03C8   982
          51    6C  D0  03C8   983         MOVL    P1(AP),R1               ; Get address of characteristics
                0C  10  03CB   984         BSBB    RTT_CHARSIZE            ; Obtain the size of the char buffer
          50    0C  3C  03CD   985         MOVZWL  #SS$_ACCVIO,R0          ; Assume access violation
                        03D0   986         IFNORD  R2,(R1),10$             ; Characteristics accessible?
                    05  03D6   987         RSB                             ; return
                        03D7   988 10$:
                    E0  11  03D7   989         BRB     ABORT                   ; error
                        03D9   990
                        03D9   991         .SBTTL  RTT_CHARSIZE, Size of characteristics buffer
                        03D9   992
                        03D9   993 RTT_CHARSIZE:
      52    04 AC   D0  03D9   994         MOVL    P2(AP), R2      ; Size of characters buffer
                0F  13  03DD   995         BEQL    10$             ; Zero is for 8
          08    52  D1  03DF   996         CMPL    R2, #8          ; 8 is allowed
                0D  13  03E2   997         BEQL    20$             ; Ok
                0C  1F  03E4   998         BLSSU   30$             ; Less is no good
                10  10  03E6   999         BSBB    RTT_ECOQ        ; If greater then we must be latest
          0C    52  D1  03E8  1000         CMPL    R2, #12         ; Must be 12 and nothing else
                05  12  03EB  1001         BNEQ    30$             ; No good
                    05  03ED  1002         RSB                     ; Ok
          52    08  D0  03EE  1003 10$:    MOVL    #8, R2          ; Use 8 if zero
                    05  03F1  1004 20$:    RSB
                        03F2  1005
          50    14  3C  03F2  1006 30$:    MOVZWL  #SS$_BADPARAM, R0       ; Abort qio with an error
             FFC1  31  03F5  1007         BRW     ABORT
                        03F8  1008
                        03F8  1009         .SBTTL  RTT_ECOQ, Validate latest eco number
                        03F8  1010 ;++
                        03F8  1011 ; RTT_ECOQ
                        03F8  1012 ;
                        03F8  1013 ; inputs
                        03F8  1014 ;       r3 -> irp
                        03F8  1015 ;       r5 -> ucb
                        03F8  1016 ; outputs
                        03F8  1017 ;       return if eco is latest,
                        03F8  1018 ;       else abort QIO with ss$_badparam
                        03F8  1019 ;--
                        03F8  1020
                        03F8  1021 RTT_ECOQ:
        40 A3   B4  03F8  1022         CLRW    IRP$W_RTT_COMPAT(R3)    ; Make sure its zero
      00D5 C5   95  03FB  1023         TSTB    UCB$B_RTT_PROECO(R5)    ; Latest for now is just a one
```

```
                                                B 2
                     06   12  03FF  1024          BNEQ    10$                     ; zero is last eco level
               0699  8F   B0  0401  1025          MOVW    #SS$_INCOMPAT,-         ; Return quiet error
                 40  A3       0405  1026                  IRP$Q_RTT_COMPAT(R3)    ; message
                     05  0407  1027 10$:    RSB
```

```
                     0408  1029          .SBTTL  RTT_SENSEMODE, Function Decision Routine for SENSEMODE/SENSECHAR
                     0408  1030 ;++
                     0408  1031 ; RTT_SENSEMODE, Function Decision Routine for SENSEMODE/SENSECHAR Functions
                     0408  1032 ;
                     0408  1033 ; Functional description:
                     0408  1034 ;
                     0408  1035 ;       This routine is called by the SYS$QIO service to dispatch a SENSEMODE
                     0408  1036 ;       or SENSECHAR I/O request.
                     0408  1037 ;
                     0408  1038 ;       The QIO parameters for terminal SENSEMODE/SENSECHAR are:
                     0408  1039 ;
                     0408  1040 ;       P1 = address of 8 or 12 byte characteristics buffer
                     0408  1041 ;       P2 = 0, 8 or 12
                     0408  1042 ;
                     0408  1043 ;       The buffer is validated for access, the process's quota checked and
                     0408  1044 ;       decremented, a message block is allocated, the address of the message
                     0408  1045 ;       block is stored in the IRP, and the IRP is queued to the ACP for
                     0408  1046 ;       delivery to the remote system.
                     0408  1047 ;
                     0408  1048 ; Inputs:
                     0408  1049 ;
                     0408  1050 ;       R0-R2 = scratch registers
                     0408  1051 ;       R3 = address of the IRP (I/O request packet)
                     0408  1052 ;       R4 = address of the PCB (process control block)
                     0408  1053 ;       R5 = address of the UCB (unit control block)
                     0408  1054 ;       R6 = address of the CCB (channel control block)
                     0408  1055 ;       R7 = bit number of the I/O function code
                     0408  1056 ;       R8 = address of the FDT table entry for this routine
                     0408  1057 ;       R9-R11 = scratch registers
                     0408  1058 ;       AP = address of the 1st function dependent QIO parameter
                     0408  1059 ;
                     0408  1060 ; Outputs:
                     0408  1061 ;
                     0408  1062 ;       IRP$L_SVAPTE(R3) = address of message buffer
                     0408  1063 ;       IRP$W_BOFF(R3) = size of message buffer
                     0408  1064 ;       IRP$L_MEDIA(R3) = address of user characteristics buffer
                     0408  1065 ;       IRP$W_BCNT(R3) = size of user characteristics buffer, 8
                     0408  1066 ;
                     0408  1067 ;       The routine preserves all registers except R0-R2, and R9-R11
                     0408  1068 ;--
                     0408  1069 RTT_SENSEMODE:                                  ; SENSEMODE/SENSECHAR FDT routine
          40 A3   B4 0408  1070          CLRW    IRP$W_RTT_COMPAT(R3)        ; No compatibility error
                     040B  1071
     59   20 A3   3C 040B  1072          MOVZWL  IRP$W_FUNC(R3),R9           ; Fetch function code
   08 59    07   E1 040F  1073          BBC     #IO$V_RD_MODEM,R9,5$        ; skip if not read modem
50   0334 8F   3C 0413  1074          MOVZWL  #SS$_DEVREQERR, R0          ; Return as device request error
        FF9E   31 0418  1075          BRW     ABORT                       ; with an error not success
                     041B  1076 5$:
     51   6C   D0 041B  1077          MOVL    P1(AP),R1                   ; Get address of characteristics buffer
        FFB8   30 041E  1078          BSBW    RTT_CHARSIZE                ; Size of chars buffer (return in R2)
     50   0C   3C 0421  1079          MOVZWL  #SS$_ACCVIO,R0              ; Assume access violation
                     0424  1080          IFWRT   R2,(R1),10$                 ; Buffer accessible?
        FF8C   31 042A  1081 7$:       BRW     ABORT                       ; Branch if not
                     042D  1082 10$:
   08 59    0E   E1 042D  1083          BBC     #IO$V_BRDCST,R9,15$         ; Branch if not brdcst bit request
61   00A8 C5   7D 0431  1084          MOVQ    UCB$Q_TL_BRKTHRU(R5),(R1) ; read bits (no remoting of this?)
        FF86   31 0436  1085          BRW     FDT_FINISHIOC_OK            ; Complete I/O
```

```
                     0439  1086 15$:
      51    6C   D0  0439  1087          MOVL    P1(AP),R1                       ; Get address of characteristics buffer
            9B   10  043C  1088          BSBB    RTT_CHARSIZE                    ; Size of chars buffer
      50    0C   3C  043E  1089          MOVZWL  #SS$_ACCVIO,R0                  ; Assume access violation
                     0441  1090          IFNOWRT R2,(R1),7$                      ; Buffer accessible?
      00D5  C5   95  0447  1091          TSTB    UCB$B_RTT_PROECO(R5)            ; Previous version
            12   12  044B  1092          BNEQ    20$                             ; Nope
            3F   AB  044D  1093          BICW3   #IRP$M_FCODE,-                  ; Obtain the modifiers
   50    20   A3        044F  1094                  IRP$W_FUNC(R3), R0              ; to look for bad ones
0040 8F    50   B1  0452  1095          CMPW    R0, #IO$M_TYPEAHDCNT            ; Only good one
            06   13  0457  1096          BEQL    20$                             ; Ok
      0699  8F   B0  0459  1097          MOVW    #SS$_INCOMPAT,-                 ; Return quiet error
      40    A3        045D  1098                  IRP$W_RTT_COMPAT(R3)            ; to signal the incompatibility
   38 A3    51   D0  045F  1099 20$:     MOVL    R1,IRP$L_MEDIA(R3)              ; Save address in packet
   32 A3    52   B0  0463  1100          MOVW    R2,IRP$W_BCNT(R3)               ; Set size in packet
   2A A3    02   A8  0467  1101          BISW    #IRP$M_FUNC,IRP$W_STS(R3)       ; Set READ type function
      51    18   D0  046B  1102          MOVL    #RBF$K_HEADERLEN,R1             ; Set size of message buffer
         000D   30  046E  1103          BSBW    ALLOC_MESSAGE                   ; Allocate the message buffer
   52    18 A2   9E  0471  1104          MOVAB   RBF$L_PARAM1(R2),R2             ; R2 points to end of data
         0303   31  0475  1105          BRW     RTT_NETMSGSENDX                 ; Send the message and exit service
```

```
                                                E 2
0478  1107          .SBTTL  ALLOC_MESSAGE, Allocate a message buffer
0478  1108 ;++
0478  1109 ; ALLOC_MESSAGE, Allocate a message buffer to send to remote process
0478  1110 ; SET_MSGHDR,    Setup a message header for broadcast
0478  1111 ;
0478  1112 ; Functional description:
0478  1113 ;
0478  1114 ;       This routine checks that the process has sufficient buffered I/O
0478  1115 ;       byte count quota for the message buffer, and then allocates the
0478  1116 ;       buffer from non-paged pool.  The process's buffered I/O byte count
0478  1117 ;       quota is decreased by the size of the allocated buffer and the
0478  1118 ;       message header information is stored.
0478  1119 ;
0478  1120 ; Inputs:
0478  1121 ;
0478  1122 ;       R1 = size of message required
0478  1123 ;       R3 = address of IRP
0478  1124 ;       R4 = address of PCB
0478  1125 ;
0478  1126 ; Outputs:
0478  1127 ;
0478  1128 ;       R1 = size of buffer
0478  1129 ;       R2 = address of buffer
0478  1130 ;
0478  1131 ;       IRP$L_SVAPTE(R3) = address of buffer
0478  1132 ;       IRP$W_BOFF(R3) = size of buffer
0478  1133 ;
0478  1134 ;       RBF$B_TYPE(R2)  = Block type
0478  1135 ;       RBF$W_SIZE(R2) = size of message buffer
0478  1136 ;       RBF$W_OPCODE(R2) = I/O function
0478  1137 ;       RBF$W_MOD(R2) = I/O function modifiers
0478  1138 ;       RBF$L_REFID(R2) = Reference id of function
0478  1139 ;       RBF$W_UNIT(R2) = Set to SVPN of the ucb (?? not used really)
0478  1140 ;
0478  1141 ;       If process does not have sufficient quota, the I/O request
0478  1142 ;       is aborted.
0478  1143 ;--
0478  1144 ALLOC_ABORT:
      53 8ED0 0478  1145          POPL    R3                      ; Restore IRP
    FF3B   31 047B  1146          BRW     ABORT                   ; and abort the I/O
047E  1147
047E  1148 ALLOC_MESSAGE:                                         ; Allocate message buffer
      53   DD 047E  1149          PUSHL   R3                      ; Save packet address
00000000'GF   16 0480  1150          JSB     G^EXE$BUFFRQUOTA        ; Check quota
      EF 50  E9 0486  1151          BLBC    R0,ALLOC_ABORT          ; Branch if error
0489  1152 ;
0489  1153 ; Allocate the message buffer
0489  1154 ;
00000000'GF   16 0489  1155          JSB     G^EXE$ALLOCBUF          ; Allocate the buffer
      E6 50  E9 048F  1156          BLBC    R0,ALLOC_ABORT          ; Branch if error
      53 8ED0 0492  1157          POPL    R3                      ; Restore packet address
0495  1158 ;
0495  1159 ; Adjust process's quota
0495  1160 ;
50   0080 C4   D0 0495  1161          MOVL    PCB$L_JIB(R4),R0        ; Get Job Information Block address
  20 A0   51   C2 049A  1162          SUBL    R1,JIB$L_BYTCNT(R0)     ; Adjust buffered I/O byte count quota
  30 A3   51   B0 049E  1163          MOVW    R1,IRP$W_BOFF(R3)       ; Save buffer size as quota
```

```
                                04A2  1165 ;
                                04A2  1166 ; Store message header information
                                04A2  1167 ;
                                04A2  1168
                                04A2  1169 ;
                                04A2  1170 ;       R0      = Clobbered
                                04A2  1171 ;       R1      = Buffer size
                                04A2  1172 ;       R2      = Buffer address
                                04A2  1173 ;       R3      = IRP address
                                04A2  1174 ;
                                04A2  1175
                                04A2  1176 SET_MSGHDR:
                                04A2  1177
      12 A2    50 A3    D0      04A2  1178          MOVL    IRP$L_SEQNUM(R3), -     ; Sequence number of the operation
                                04A7  1179                  RBF$L_REFID(R2)
         50    1C A3    D0      04A7  1180          MOVL    IRP$L_UCB(R3), R0       ; Unit control block address
      16 A2    74 A0    B0      04AB  1181          MOVW    UCB$L_SVPN(R0), -       ; Bogus unit number, not used
                                04B0  1182                  RBF$W_UNIT(R2)
                  00    EF      04B0  1183          EXTZV   #IRP$V_FCODE,-          ; Set requested function code
   0E A2    20 A3    06          04B2  1184                  #IRP$S_FCODE,IRP$W_FUNC(R3),RBF$W_OPCODE(R2)
   10 A2    20 A3    3F    AB    04B7  1185          BICW3   #IRP$M_FCODE,IRP$W_FUNC(R3),RBF$W_MOD(R2) ; Set requested modifiers
                                04BD  1186
                                04BD  1187 ;
                                04BD  1188 ; Setup a message header but don't depend on the irp address
                                04BD  1189 ; except for svapte.
                                04BD  1190 ;
                                04BD  1191
                                04BD  1192 SET_MSGHDRX:
                                04BD  1193
   2C A3    52    D0      04BD  1194          MOVL    R2,IRP$L_SVAPTE(R3)     ; Save buffer address in packet
   08 A2    51    B0      04C1  1195          MOVW    R1,RBF$W_SIZE(R2)       ; Save buffer size in message
            13    90      04C5  1196          MOVB    #DYN$C_BUFIO,-          ; Set block type
         0A A2              04C7  1197                  RBF$B_TYPE(R2)
         0E A2    9E      04C9  1198          MOVAB   RBF$W_OPCODE(R2),-      ; Set address of data
                  62      04CC  1199                  RBF$L_MSGDAT(R2)
         04 A2    D4      04CD  1200          CLRL    RBF$L_USRBFR(R2)        ; Set user buffer address
                  05      04D0  1201          RSB                             ;
```

```
                                04D1  1203          .SBTTL  RTT_INTERRUPT Interrupt handler
                                04D1  1204 ;++
                                04D1  1205 ; RTT_INTERRUPT, I/O completion interrupt handler
                                04D1  1206 ;
                                04D1  1207 ; Functional description:
                                04D1  1208 ;
                                04D1  1209 ;       This routine handles an I/O completion "interrupt" from the ACP.
                                04D1  1210 ;       The I/O status and data is obtained from the response packet from
                                04D1  1211 ;       the remote terminal handler process, and the I/O request is completed.
                                04D1  1212 ;
                                04D1  1213 ; Inputs:
                                04D1  1214 ;
                                04D1  1215 ;       R3 = address of the IRP
                                04D1  1216 ;       R5 = address of UCB
                                04D1  1217 ;       IRP$L_SVAPTE(R3) = address of response message
                                04D1  1218 ;
                                04D1  1219 ;       IPL = 0
                                04D1  1220 ;
                                04D1  1221 ; Outputs:
                                04D1  1222 ;
                                04D1  1223 ;       I/O status copied to IRP$L_IOST and I/O request posted.
                                04D1  1224 ;
                                04D1  1225 ;       This routine only needs to preserve R11.
                                04D1  1226 ;--
                                04D1  1227 RTT_INTERRUPT:                               ; I/O completion interrupt handler
52   2C A3    D0                04D1  1228          MOVL    IRP$L_SVAPTE(R3),R2        ; Get address of message
   51   62    D0                04D5  1229          MOVL    (R2),R1                    ; Address of data in buffer
        01    E1                04D8  1230          BBC     #IRP$V_FUNC,-              ; If clr not READ/SENSE/BROADCAST
 47 2A A3                       04DA  1231                  IRP$W_STS(R3),POST
        00    EF                04DD  1232          EXTZV   #IRP$V_FCODE,-             ; Get original function code
        06                      04DF  1233                  #IRP$S_FCODE,-
50   20 A3                      04E0  1234                  IRP$W_FUNC(R3),R0
        60    13                04E3  1235          BEQL    POST_BROADCAST             ; If eql BROADCAST function
   27   50    91                04E5  1236          CMPB    R0,#IO$_SENSEMODE          ; SENSEMODE function?
        1C    13                04E8  1237          BEQL    POST_SENSE                 ; If eql yes
   1B   50    91                04EA  1238          CMPB    R0,#IO$_SENSECHAR          ; SENSECHAR function?
        17    13                04ED  1239          BEQL    POST_SENSE                 ;If eql yes - else read function
                                04EF  1240 ;
                                04EF  1241 ; Set up buffer to post READ
                                04EF  1242 ;
62   14 A1    9E                04EF  1243          MOVAB   RDP$T_TT_RDATA+2(R1),(R2) ; Set address of data
04 A2   38 A3    D0            04F3  1244          MOVL    IRP$L_MEDIA(R3),4(R2)      ; Set address of user buffer
   12 A1    B1                  04F8  1245          CMPW    RDP$T_TT_RDATA(R1),-       ; Size of data greater than user buffer?
   32 A3                        04FB  1246                  IRP$W_BCNT(R3)             ;
   25       1A                  04FD  1247          BGTRU   POST                       ; If gtru yes - leave user's size
   12 A1    B0                  04FF  1248          MOVW    RDP$T_TT_RDATA(R1),-       ; Else, set size to actual data size
   32 A3                        0502  1249                  IRP$W_BCNT(R3)             ;
      1E    11                  0504  1250          BRB     POST                       ;
                                0506  1251 ;
                                0506  1252 ; Set up buffer to post SENSEMODE/CHAR
                                0506  1253 ;
                                0506  1254 POST_SENSE:                                ;
                                0506  1255 ;
                                0506  1256 ;       Note that for the latest protocol, either 8 or 12 bytes will come
                                0506  1257 ;       from this part of the message.  Size is already in IRP.
                                0506  1258 ;
62   12 A1    9E                0506  1259          MOVAB   RDP$Q_TT_SCHAR(R1),(R2) ; Set address of data
```

```
                04 A2    38 A3   D0  050A  1260          MOVL    IRP$L_MEDIA(R3),4(R2)   ; Set address of user data
                      00D5 C5   95  050F  1261          TSTB    UCB$B_RTT_PROECO(R5)    ; Latest version
                           05   12  0513  1262          BNEQ    10$                     ; Yes
                         48 A5   D0  0515  1263          MOVL    UCB$L_DEVDEPND2(R5),-   ; Return additional characters if
                         1A A1       0518  1264                  RDP$L_TT_SCHAR2(R1)     ; they are requested
                                     051A  1265 10$:
                      FFC0 8F    B3  051A  1266          BITW    #^CIRP$M_FCODE,-        ; Check for spawn bits only if no
                         20 A3       051E  1267                  IRP$W_FUNC(R3)          ; modifier on the sensemode
                            02   12  0520  1268          BNEQ    20$                     ; We have modifiers
                            26   10  0522  1269          BSBB    SENSE_SPAWN             ; Set the three bits for spawn
                                     0524  1270 20$:
                                     0524  1271 POST:                                  ; Post the I/O
              2A A3   0200 8F    A8  0524  1272          BISW    #IRP$M_TERMIO,IRP$W_STS(R3) ; Set terminal I/O completion
                         0A A1   7D  052A  1273          MOVQ    RDP$Q_STATUS(R1),-      ; Set I/O status
                         38 A3       052D  1274                  IRP$L_IOST1(R3)         ;
                         38 A3   B1  052F  1275          CMPW    IRP$L_IOST1(R3),-       ; If normal return
                            01       0532  1276                  #SS$_NORMAL             ;
                            0A   12  0533  1277          BNEQ    10$                     ; Nope
                         40 A3   B5  0535  1278          TSTW    IRP$W_RTT_COMPAT(R3)    ; Check for compatibility error
                            05   13  0538  1279          BEQL    10$                     ; Nope
                         40 A3   B0  053A  1280          MOVW    IRP$W_RTT_COMPAT(R3),-  ; Return compatibility error
                         38 A3       053D  1281                  IRP$L_IOST1(R3)         ; to user
                   00000000'GF   17  053F  1282 10$:     JMP     G^COM$POST              ; Post the I/O
                                     0545  1283 ;
                                     0545  1284 ; Post a BROADCAST completion
                                     0545  1285 ;
                                     0545  1286 POST_BROADCAST:                         ;
                                     0545  1287          BUG_CHECK BRDMSGLOST            ; NOT supposed to get here...
                                  05 0549  1288          RSB
```

```
                        054A  1290          .SBTTL  SENSE_SPAWN  Sense for spawn
                        054A  1291
                        054A  1292 ;       Sense special characteristics bits for DCL spawn command.
                        054A  1293 ;       Return bits for ctrl/c ast, outofband ast and associated mailbox.
                        054A  1294 ;       These bits may be reused later and are not for customer application
                        054A  1295 ;       consumption.
                        054A  1296 ;
                        054A  1297 ; inputs:
                        054A  1298 ;       r1 -> RDP message
                        054A  1299 ;
                        054A  1300
                        054A  1301 SENSE_SPAWN:
   50   1A A1   9E      054A  1302          MOVAB   RDP$L_TT_SCHAR2(R1), R0 ; Address of the characteristics
60    0200 8F   AA      054E  1303          BICW    #TT2$M_DCL_MAILBX,(R0)  ; Reset
      60 A5     D5      0553  1304          TSTL    UCB$L_AMB(R5)           ; Any associated mailbox?
         05     13      0556  1305          BEQL    10$                     ; No
60    0200 8F   A8      0558  1306          BISW    #TT2$M_DCL_MAILBX,(R0)  ; Yes, so set characteristic
                        055D  1307 10$:
                     05 055D  1308          RSB
```

```
                     055E  1310          .SBTTL  RTT_CANCEL, Cancel I/O routine
                     055E  1311 ;++
                     055E  1312 ; RTT_CANCEL, Cancels an I/O operation in progress
                     055E  1313 ;
                     055E  1314 ; Functional description:
                     055E  1315 ;
                     055E  1316 ;       This routine cancels any CTRL/C or CTRL/Y AST's that were
                     055E  1317 ;       requested by the cancelling process on the cancelling channel.
                     055E  1318 ;
                     055E  1319 ;       If there are no more references remaining to the device, the UCB
                     055E  1320 ;       is queued to the ACP to notify it that the device is no longer in
                     055E  1321 ;       use.  The ACP will then check that the reference count is still zero
                     055E  1322 ;       and remove the UCB from I/O database and deallocate it.
                     055E  1323 ;
                     055E  1324 ; Inputs:
                     055E  1325 ;
                     055E  1326 ;       R2 = negated value of the channel index number
                     055E  1327 ;       R3 = address of the current IRP (I/O request packet)
                     055E  1328 ;       R4 = address of the PCB (process control block) for the
                     055E  1329 ;               process canceling I/O
                     055E  1330 ;       R5 = address of the UCB (unit control block)
                     055E  1331 ;
                     055E  1332 ;       IPL = driver fork IPL
                     055E  1333 ;
                     055E  1334 ; Outputs:
                     055E  1335 ;
                     055E  1336 ;       DEV$M_DMT is set in UCB$L_DEVCHAR to prevent a race if someone
                     055E  1337 ;       assigns and deassigns another channel to the UCB before the ACP
                     055E  1338 ;       dequeues the UCB.
                     055E  1339 ;
                     055E  1340 ;       The routine preserves all registers except R0-R3.
                     055E  1341 ;--
                     055E  1342 .ENABLE LOCAL_BLOCK
                     055E  1343
                     055E  1344 ASSUME  CAN$C_CANCEL EQ 0
                     055E  1345 ASSUME  CAN$C_DASSGN EQ 1
                     055E  1346
          00A4   31  055E  1347 10$:    BRW     50$
          009E   31  0561  1348 20$:    BRW     40$
                     0564  1349
                     0564  1350 RTT_CANCEL:                                ; Cancel an I/O operation
      00F0 8F    BB  0564  1351         PUSHR   #^M<R4,R5,R6,R7>           ; Save registers
           04    E1  0568  1352         BBC     #UCB$V_ONLINE,-            ; If clr unit offline - probably template
      F1 64 A5       056A  1353                 UCB$W_STS(R5),10$          ;
         5C A5    B5  056D  1354         TSTW    UCB$W_REFC(R5)             ; Any more references to device?
           EF    13  0570  1355         BEQL    20$                        ; Nope all done.
                     0572  1356
         56   52    D0  0572  1357         MOVL    R2,R6                      ; Make a copy of channel number
              58    D5  0575  1358         TSTL    R8                         ; Cancel or deassign
              0B    13  0577  1359         BEQL    25$                        ; Cancel
                     0579  1360
    57   0090 C5    DE  0579  1361         MOVAL   UCB$L_RTT_CTRLY(R5),R7     ; Get address of CTRL/Y AST list
     00000000'GF    16  057E  1362         JSB     G^COM$FLUSHATTNS           ; Flush all cancelled AST's
                     0584  1363
    57   0094 C5    DE  0584  1364 25$:    MOVAL   UCB$L_RTT_CTRLC(R5),R7     ; Get address of CTRL/C AST list
     00000000'GF    16  0589  1365         JSB     G^COM$FLUSHATTNS           ; Flush any cancelled AST's
    57   00C4 C5    9E  058F  1366         MOVAB   UCB$L_RTT_BANDINCL(R5), R7 ; Flush any outofband asts
```

```
                52   00C8 C5    9E   0594  1367          MOVAB   UCB$L_RTT_BANDINMSK(R5), R2 ; mask address
                 00000000'GF    16   0599  1368          JSB     G^COM$FLUSHCTRLS           ; Flush them by channel etc
                57   009C C5    9E   059F  1369          MOVAB   UCB$L_RTT_BANDEXCL(R5), R7 ; Flush any outofband asts
                52   0098 C5    9E   05A4  1370          MOVAB   UCB$L_RTT_BANDEXMSK(R5), R2 ; mask address
                 00000000'GF    16   05A9  1371          JSB     G^COM$FLUSHCTRLS           ; Flush them by channel etc
                                    05AF  1372 ;
                                    05AF  1373 ; If we are talking to new version, tell him the new masks.
                                    05AF  1374 ;
                     00D5 C5    95   05AF  1375          TSTB    UCB$B_RTT_PROECO(R5)       ; Nonzero for latest
                          48    13   05B3  1376          BEQL    30$                        ; Old version
                          22    D0   05B5  1377          MOVL    #RBF$B_TT_OUTBAND+1+4+1+4,- ; Size of the outband message
                          51         05B7  1378                  R1                         ; buffer
                          53    DD   05B8  1379          PUSHL   R3                         ; Save across dirty routine
                00000000'GF    16   05BA  1380          JSB     G^EXE$ALONONPAGED          ; Get me some memory
                          53 8ED0   05C0  1381          POPL    R3                         ; restore packet address
                       37 50    E9   05C3  1382          BLBC    R0, 30$                    ; Hang it up for lack of space?
                                    05C6  1383
                                    05C6  1384 ;
                                    05C6  1385 ; Here comes an incredible hack.  We are going to build a message to be
                                    05C6  1386 ; transmitted which has no irp context.  It will have a REFID of zero.
                                    05C6  1387 ; To do this we need an irp address with a svapte field to save the
                                    05C6  1388 ; packet address.  We make an "irp" by passing the address of a cell in
                                    05C6  1389 ; the ucb which can be used.  The address is backed up by the svapte offset
                                    05C6  1390 ; so that for this purpose it looks like an irp.
                                    05C6  1391 ;
                                    05C6  1392
                          53    DD   05C6  1393          PUSHL   R3                         ; Save the bad r3
                53   4C A5    9E   05C8  1394          MOVAB   <UCB$L_SVAPTE - -          ; Make a bogus irp address
                                    05CC  1395                  IRP$L_SVAPTE>(R5), R3      ; with only a good svapte
                           FEEE   30   05CC  1396          BSBW    SET_MSGHDRX                ; Set up the message header
                        12 A2     D4   05CF  1397          CLRL    RBF$L_REFID(R2)            ; Ref id is zero
                        16 A2     B4   05D2  1398          CLRW    RBF$W_UNIT(R2)             ; No unit specified
                           14     B0   05D5  1399          MOVW    #RDP$B_TT_OUTBAND+1+4+1+4,- ; Size of data to be sent
                        0C A2         05D7  1400                  RBF$W_DATSIZE(R2)          ; to the server
                     0E A2    23     B0   05D9  1401          MOVW    #IO$_SETMODE, -            ; Set the op
                                    05DD  1402                  RBF$W_OPCODE(R2)           ; code of the message
                10 A2   0400 8F     B0   05DD  1403          MOVW    #IO$M_OUTBAND, -           ; and the modifier
                                    05E3  1404                  RBF$W_MOD(R2)              ; for the message
                     52    18 A2    9E   05E3  1405          MOVAB   RBF$B_TT_OUTBAND(R2), R2; Now build the message itself
                          82    04   90   05E7  1406          MOVB    #4, (R2)+                  ; Count for include mask
                          00C8 C5    D0   05EA  1407          MOVL    UCB$L_RTT_BANDINMSK(R5),-; Include mask
                               82         05EE  1408                  (R2)+                      ;
                          82    04   90   05EF  1409          MOVB    #4, (R2)+                  ; Count for exclude mask
                          0098 C5    D0   05F2  1410          MOVL    UCB$L_RTT_BANDEXMSK(R5),- ; Now the exclude mask
                               82         05F6  1411                  (R2)+                      ;
                             01A4   30   05F7  1412          BSBW    RTT_NETCANSEND             ; Send the message to the server
                               53 8ED0   05FA  1413          POPL    R3                         ; Restore the bogus irp address
                                    05FD  1414 30$:
                                    05FD  1415
                             02E7   30   05FD  1416          BSBW    RTT_CANIRPS                ; Cancel outstanding IRPs
                               03   11   0600  1417          BRB     50$
                                    0602  1418
                                    0602  1419 40$:
                                    0602  1420 ;
                                    0602  1421 ; Clean up the ucb after all references have gone
                                    0602  1422 ;
                             0118   30   0602  1423          BSBW    RTT_ABORTIRPS              ; Flush all irps from queue
```

```
                     0605  1424
                     0605  1425 50$:
00F0 8F    BA        0605  1426          POPR    #^M<R4,R5,R6,R7>        ; Restore registers
           05        0609  1427          RSB                             ; Return
                     060A  1428
                     060A  1429 .DISABLE LOCAL_BLOCK
```

```
                        060A  1431          .SBTTL  RTT_UNSOLIC Unsolicited interrupt handler
                        060A  1432 ;++
                        060A  1433 ; RTT_UNSOLIC, Unsolicted interrupt handler
                        060A  1434 ;
                        060A  1435 ; Functional description:
                        060A  1436 ;
                        060A  1437 ;       This routine handles unsolicted attention messages from the remote
                        060A  1438 ;       terminal handler process.  If the message is:
                        060A  1439 ;
                        060A  1440 ;               Unsolicited data: If device has any references, deliver message
                        060A  1441 ;                                 to associated mailbox; if no references,
                        060A  1442 ;                                 deliver a message to the Job Controller.
                        060A  1443 ;
                        060A  1444 ;               Hang-up:          Deliver any CNTRL/Y AST's, specifying hang-up;
                        060A  1445 ;                                 deliver a hangup message to associated mailbox.
                        060A  1446 ;
                        060A  1447 ;               CTRL/C or CTRL/Y: Any corresponding AST's are delivered.
                        060A  1448 ;
                        060A  1449 ;               STARTRCV         Start the receive to the net.
                        060A  1450 ;
                        060A  1451 ; Inputs:
                        060A  1452 ;
                        060A  1453 ;       R3 = address of attention message
                        060A  1454 ;       R5 = address of UCB
                        060A  1455 ;
                        060A  1456 ;       IPL = 0
                        060A  1457 ;
                        060A  1458 ; Outputs:
                        060A  1459 ;
                        060A  1460 ;       Message or AST(s) delivered and attention message block deallocated.
                        060A  1461 ;
                        060A  1462 ;--
                        060A  1463 RTT_UNSOLIC:                                 ; Unsolicted interrupt handler
              53    DD  060A  1464          PUSHL   R3                        ; Save address of mesage block
                    DD  060C  1465          DSBINT  UCB$B_FIPL(R5)            ; Synchronize access to UCB
          51    63    D0  0613  1466          MOVL    (R3),R1                   ; Obtain the address of the data
                        0616  1467          CASE    RDP$W_MOD(R1),<-          ; Case on message modifier type
                        0616  1468                  UNSOL_DATA,-              ;  Unsolicited data
                        0616  1469                  HANGUP,-                  ;  Hangup
                        0616  1470                  CTRLC,-                   ;  CNTRL/C
                        0616  1471                  CTRLY,-                   ;  CNTRL/Y
                        0616  1472                  STARTRCV,-                 ;  Start network receive
                        0616  1473                  RTT_BRDCST,-              ; Broadcast message for mailbox
                        0616  1474                  RTT_OUTBAND,-             ; Out of band ast character
                        0616  1475                  >,LIMIT=#RBF$C_TT_UNSOL   ;
            0050    31  0629  1476          BRW     UNSOLIC_EXIT              ;
                        062C  1477 ;
                        062C  1478 ; Deliver unsolicited data notification
                        062C  1479 ;
                        062C  1480 UNSOL_DATA:                                  ; Unsolicited data
          54    01    9A  062C  1481          MOVZBL  #MSG$_TRMUNSOLIC,R4       ; Set mailbox message type
            5C  A5    B5  062F  1482          TSTW    UCB$W_REFC(R5)            ; Any references to device?
                11    13  0632  1483          BEQL    10$                       ; If eql no - notify Job Controller
      53    60  A5    D0  0634  1484          MOVL    UCB$L_AMB(R5),R3          ; Get address of associated mailbox
                24    13  0638  1485          BEQL    20$                       ; If eql none - forget it
00000000'GF     16  063A  1486          JSB     G^EXE$SNDEVMSG            ; Deliver notification to mailbox
                1B  50    E9  0640  1487          BLBC    R0,20$                    ; If lbc failure
```

```
                     19   11  0643  1488                BRB     20$                             ;
                                0645  1489 10$:
                53 00000000'GF   D0  0645  1490                MOVL    G^TTY$GL_JOBCTLMB,R3    ; Get address of Job Controller mailbox
                   0D 68 A5      00  E0  064C  1491                BBS     #UCB$V_JOB,UCB$W_DEVSTS(R5),20$ ; Branch if notified already
                   00000000'GF   16  0651  1492                JSB     G^EXE$SNDEVMSG          ; Deliver notification to mailbox
                         04 50   E9  0657  1493                BLBC    R0,20$                  ; If lbc failure
                      68 A5 01   A8  065A  1494                BISW    #UCB$M_JOB,UCB$W_DEVSTS(R5) ; Set Job Controller notified
                                0065E 1495 20$:
                     1C   11  065E  1496                BRB     UNSOLIC_EXIT            ;
                                0660  1497
                                0660  1498 ;
                                0660  1499 ; Deliver hangup notification
                                0660  1500 ;
                                0660  1501
                                0660  1502 HANGUP:                                  ; Dataset hangup
                   008C   30    0660  1503                BSBW    RTT_HANGUP              ; Do the hangup stuff
                     17   11    0663  1504                BRB     UNSOLIC_EXIT            ;
                                0665  1505
                                0665  1506 ;
                                0665  1507 ; Start network receive
                                0665  1508 ;
                                0665  1509
                                0665  1510 STARTRCV:
                   0197   30    0665  1511                BSBW    RTT_STARTNETRCV         ; Start it out of line
                     12   11    0668  1512                BRB     UNSOLIC_EXIT
                                066A  1513
                                066A  1514 ;
                                066A  1515 ; Deliver any CNTRL/C AST's
                                066A  1516 ;
                                066A  1517
                                066A  1518 CTRLC:                                  ; Deliver CNTRL/C AST's
54                 0094 C5      DE  066A  1519                MOVAL   UCB$L_RTT_CTRLC(R5),R4  ; Get address of CNTRL/C AST list
                     05   11    066F  1520                BRB     DELAST                  ;
                                0671  1521
                                0671  1522 ;
                                0671  1523 ; Deliver any CNTRL/Y AST'S
                                0671  1524 ;
                                0671  1525
                                0671  1526 CTRLY:                                  ; Deliver CNTRL/Y AST's
54                 0090 C5      DE  0671  1527                MOVAL   UCB$L_RTT_CTRLY(R5),R4  ; Get address of CNTRL/Y AST list
                                0676  1528 DELAST:                                 ;
 00000000'GF                     16  0676  1529                JSB     G^COM$DELATTNAST        ; Deliver the AST's
                                067C  1530                                          ;
                                067C  1531 UNSOLIC_EXIT:                           ; Exit unsolicited message handler
                                067C  1532                ENBINT                  ; Re-enable interrupts
                         50 8ED0      067F  1533                POPL    R0                      ; Get address of message block
                   0A A0    13   90  0682  1534                MOVB    #DYN$C_BUFIO,IRP$B_TYPE(R0) ; Be sure buffer type is valid
                00000000'GF   16   16  0686  1535                JSB     G^EXE$DEANONPAGED       ; Deallocate the message block
                              05   068C  1536                RSB
```

```
                                068D  1538
                                068D  1539 ;+
                                068D  1540 ; RTT_BRDCST
                                068D  1541 ;
                                068D  1542 ; Deliver broadcast message to the mailbox.
                                068D  1543 ;
                                068D  1544 ; The unit number and name of the device is fixed up in the packet first.
                                068D  1545 ;
                                068D  1546 ;-
                                068D  1547
                                068D  1548 RTT_BRDCST:
                                068D  1549
            38 48 A5    04  E1  068D  1550          BBC     #TT2$V_BRDCSTMBX, -                     ; If we are allowing mailbox
                                0692  1551                  UCB$L_DEVDEPND2(R5),10$                 ; to receive the messages
                    60 A5   D5  0692  1552          TSTL    UCB$L_AMB(R5)                           ; and we have a mailbox
                       33   13  0695  1553          BEQL    10$                                     ; Nope
            0E A1   54 A5   B0  0697  1554          MOVW    UCB$W_UNIT(R5), -                       ; Then fix the unit number
                                069C  1555                  RDP$W_TT_BRDUNIT(R1)                    ; in the message
                52  28 A5   D0  069C  1556          MOVL    UCB$L_DDB(R5), R2                       ; and get the proper name of
        50  14 A2    04      00  EF  06A0  1557          EXTZV   #0, #4, DDB$T_NAME(R2), R0          ; this device for the message
                            50  D6  06A6  1558          INCL    R0                                  ; including the count
                                06A8  1559
                            3F  BB  06A8  1560          PUSHR   #^M<R0, R1, R2, R3, R4, R5>         ; Copy the new name and
 10 A1   10   00   14 A2    50  2C  06AA  1561          MOVC5   R0, DDB$T_NAME(R2), #0, -           ; clobber the remainder of the
                                06B2  1562                  #RDP$C_TT_BRDNAME, -                    ; stuff in the fixed length
                                06B2  1563                  RDP$T_TT_BRDNAME(R1)                    ; field
                            3F  BA  06B2  1564          POPR    #^M<R0, R1, R2, R3, R4, R5>         ; restore the regs
                                06B4  1565
                            38  BB  06B4  1566          PUSHR   #^M<R3, R4, R5>                     ; Save a few
          53    0A A1       3C  06B6  1567          MOVZWL  RDP$W_TT_BRDTOTSIZE(R1), R3         ; Size of the message
          54    0C A1       9E  06BA  1568          MOVAB   RDP$W_TT_BRDMSG(R1), R4             ; Address of the message
          55    60 A5       D0  06BE  1569          MOVL    UCB$L_AMB(R5), R5                   ; Mailbox ucb address
    00000000'GF             16  06C2  1570          JSB     G^EXE$WRTMAILBOX                    ; Write the message to it
                            38  BA  06C8  1571          POPR    #^M<R3, R4, R5>                     ; and ignore the errors
                                06CA  1572
                    FFAF    31  06CA  1573  10$:    BRW     UNSOLIC_EXIT                        ; Go clean up the packet.
```

```
                     06CD  1575
                     06CD  1576 ;+
                     06CD  1577 ; RTT_OUTBAND
                     06CD  1578 ;
                     06CD  1579 ; Deliver an out of band ast
                     06CD  1580 ;-
                     06CD  1581
                     06CD  1582 RTT_OUTBAND:
  53    0A A1    9A  06CD  1583          MOVZBL  RDP$B_TT_OUTBAND(R1), R3        ; Deliver the asts (char)
              53    DD  06D1  1584          PUSHL   R3                              ; Save the character
54    00C4 C5    9E  06D3  1585          MOVAB   UCB$L_RTT_BANDINCL(R5), R4      ; List address
 00000000'GF    16  06D8  1586          JSB     G^COM$DELCTRLAST                ; Deliver the asts
              53 8ED0  06DE  1587          POPL    R3                              ; Recover the character
54    009C C5    9E  06E1  1588          MOVAB   UCB$L_RTT_BANDEXCL(R5), R4      ; List address
 00000000'GF    16  06E6  1589          JSB     G^COM$DELCTRLAST                ; Deliver the asts
            FF8D    31  06EC  1590          BRW     UNSOLIC_EXIT                    ; Thats all done
```

```
                        06EF  1592          .SBTTL  RTT_HANGUP  - Perform hangup functions
                        06EF  1593          .SBTTL  RTT_ABORTIRPS - Abort irps outstanding
                        06EF  1594 ;++
                        06EF  1595 ; RTT_HANGUP  Perform hangup functions
                        06EF  1596 ; RTT_ABORTIRPS
                        06EF  1597 ;
                        06EF  1598 ; Functional description:
                        06EF  1599 ;
                        06EF  1600 ;       Deliver any CNTRL/Y AST's, specifying hang-up;
                        06EF  1601 ;       deliver a hangup message to associated mailbox.
                        06EF  1602 ;       Post any irps outstanding with abort.
                        06EF  1603 ;       Set hangup status in device status.
                        06EF  1604 ;       The ucb is passed on to the acp if there are no more
                        06EF  1605 ;       channels open to it.
                        06EF  1606 ;       HANGUP is called by net device errors and hangup operations
                        06EF  1607 ;       from the line on the other end.
                        06EF  1608 ;       ABORTIRPS is called on net device cancels and channel deassigns.
                        06EF  1609 ;
                        06EF  1610 ; Inputs:
                        06EF  1611 ;
                        06EF  1612 ;       R5 = address of UCB
                        06EF  1613 ;
                        06EF  1614 ;
                        06EF  1615 ; Outputs:
                        06EF  1616 ;
                        06EF  1617 ;       Message or AST(s) delivered.
                        06EF  1618 ;
                        06EF  1619 ;--
                        06EF  1620 RTT_HANGUP:
54   0090 C5    DE      06EF  1621          MOVAL   UCB$L_RTT_CTRLY(R5),R4  ; Get address of CTRL/Y AST list
     50   54    D0      06F4  1622          MOVL    R4,R0                   ; Copy list address
                        06F7  1623 10$:
     50   60    D0      06F7  1624          MOVL    (R0),R0                 ; Get address of next entry
          08    13      06FA  1625          BEQL    20$                     ; If eql none
     02CC 8F    3C      06FC  1626          MOVZWL  #SS$_HANGUP,-           ; Insert new parameter for AST
       1C A0            0700  1627                  ACB$L_KAST+4(R0)        ;
          F3    11      0702  1628          BRB     10$                     ;
                        0704  1629 20$:
00000000'GF    16      0704  1630          JSB     G^COM$DELATTNAST        ; Deliver the AST's
     54   06    D0      070A  1631          MOVL    #MSG$_TRMHANGUP,R4      ; Set mailbox message type
  53 60 A5    D0      070D  1632          MOVL    UCB$L_AMB(R5),R3        ; Get associated mailbox address
          06    13      0711  1633          BEQL    30$                     ; If eql none - forget it
00000000'GF    16      0713  1634          JSB     G^EXE$SNDEVMSG          ; Deliver notification to mailbox
                        0719  1635 30$:
          08    A8      0719  1636          BISW    #UCB$M_TT_HANGUP,-      ; Save hangup status
       68 A5            071B  1637                  UCB$W_DEVSTS(R5)        ;
                        071D  1638
                        071D  1639 ;
                        071D  1640 ;       Clean up the outstanding iirp read to network so it completes
                        071D  1641 ;       without calling driver again.  Post all outstanding irps with
                        071D  1642 ;       abort.
                        071D  1643 ;
                        071D  1644
                        071D  1645 RTT_ABORTIRPS:
                        071D  1646
                        071D  1647 ;
                        071D  1648 ;       We must be at ipl 7 or above here
```

```
                        071D  1649 ;
                        071D  1650          DSBINT  UCB$B_FIPL(R5)          ; Synchronize owth other entries
                        0724  1651
                        0724  1652 ;
                        0724  1653 ;        Fix the interlock with the receive iirp so it will be deallocated
                        0724  1654 ;        when it completes.  We must say we did so here.  The condition is
                        0724  1655 ;        NETIRP = 1 and IRP$L_AST = 0 means that its gone.  If NETIRP = 0
                        0724  1656 ;        it has never been allocated and given to netdriver.
                        0724  1657 ;
                        0724  1658
50   00C0 C5    D0      0724  1659          MOVL    UCB$L_RTT_NETIRP(R5),R0 ; Look at address of receive iirp
             06    13   0729  1660          BEQL    10$                     ; Nope not here
          03 50    E8   072B  1661          BLBS    R0,10$                  ; Dummy, all done?
          10 A0    D4   072E  1662          CLRL    IRP$L_AST(R0)           ; Nope so tell receive iirp
00C0 C5    01    D0      0731  1663 10$:    MOVL    #1,UCB$L_RTT_NETIRP(R5) ; Clobber address here
                        0736  1664
                        0736  1665 ;
                        0736  1666 ;        Now we abort all of the irps that we have at this time.
                        0736  1667 ;
                        0736  1668
53   00B8 D5    0F      0736  1669 20$:    REMQUE  @UCB$L_RTT_IRPFL(R5), R3; Obtain an irp from queue
             0F    1D   073B  1670          BVS     30$                     ; No more
  38 A3      2C    3C   073D  1671          MOVZWL  #SS$_ABORT, -           ; Complete with abort status
                        0741  1672                  IRP$L_IOST1(R3)         ;
       3C A3    D4      0741  1673          CLRL    IRP$L_IOST2(R3)         ;
00000000'GF    16      0744  1674          JSB     G^COM$POST              ; and poast
             EA    11   074A  1675          BRB     20$                     ; and back for more irps
                        074C  1676
                        074C  1677 ;
                        074C  1678 ;        If there are no more channels to this device, then pass it on
                        074C  1679 ;        to the acp for disposal.
                        074C  1680 ;
                        074C  1681
     5C A5      B5      074C  1682 30$:    TSTW    UCB$W_REFC(R5)          ; Any channels to device?
             26    12   074F  1683          BNEQ    50$                     ; Yes
                        0751  1684
          01    AA      0751  1685          BICW    #UCB$M_JOB,-            ; Clear Job Controller notified
       68 A5            0753  1686                  UCB$W_DEVSTS(R5)        ;
          15    E2      0755  1687          BBSS    #DEV$V_DMT,-            ; If set, UCB already queued
    1D 38 A5            0757  1688                  UCB$L_DEVCHAR(R5),50$   ;
    53    55    D0      075A  1689          MOVL    R5,R3                   ; Set up ucb as the packet
 52    34 A5    D0      075D  1690          MOVL    UCB$L_VCB(R5),R2        ; Get address of VCB
 52    10 A2    D0      0761  1691          MOVL    VCB$L_AQB(R2),R2        ; Get address of ACP AQB
00000000'GF    16      0765  1692          JSB     G^EXE$INSERTIRP         ; Insert UCB in ACP queue
             0A    12   076B  1693          BNEQ    40$                     ; If neq, not first entry in queue
 51    0C A2    D0      076D  1694          MOVL    AQB$L_ACPPID(R2),R1     ; Get ACP process ID
00000000'GF    16      0771  1695          JSB     G^SCH$WAKE              ; Wake the ACP process
                        0777  1696 40$:
                        0777  1697 50$:    ENBINT                          ; Restore IPL
                  05    077A  1698          RSB                             ;
```

```
                     077B  1700          .SBTTL RTT_NETMSGSEND  - Send message to net driver
                     077B  1701 ;
                     077B  1702 ; RTT_NETMSGSENDX  -  Send message to netdriver and exit qio
                     077B  1703 ; RTT_NETMSGSEND   -  Send message to netdriver
                     077B  1704 ; RTT_NETCANSEND   -  Send message for cancel
                     077B  1705 ; RTT_NETQUEPKT    -  Queue message to net driver
                     077B  1706 ;
                     077B  1707 ; inputs:
                     077B  1708 ;       r2 -> address beyond message data (NETMSGSEND)
                     077B  1709 ;       r3 -> rtt irp
                     077B  1710 ;       r4 -> pcb
                     077B  1711 ;       r5 -> rtt ucb
                     077B  1712 ;
                     077B  1713
                     077B  1714 RTT_NETMSGSENDX:
           06   10   077B  1715          BSBB    RTT_NETMSGSEND          ; Send the message and
00000000'GF   17   077D  1716          JMP     G^EXE$QIORETURN         ; Return from the qio
                     0783  1717
                     0783  1718 RTT_NETMSGSEND:
     50  2C A3   D0  0783  1719          MOVL    IRP$L_SVAPTE(R3),R0     ; The buffer address
           08   13  0787  1720          BEQL    10$                     ; none
51   52   60   C3  0789  1721          SUBL3   (R0),R2,R1              ; Make the length of the data
   0C A0   51   B0  078D  1722          MOVW    R1,RBF$W_DATSIZE(R0)    ; save in the buffer
      00C0 C5   E8  0791  1723 10$:     BLBS    UCB$L_RTT_NETIRP(R5),-  ; We do not have a receive posted
           3A       0795  1724                  RTT_NETHUNGUP           ; so this cannot work. We have hungup.
00BC D5   63   0E  0796  1725          INSQUE  (R3), -                 ; Queue the irp on the ucb
                     079B  1726                  @UCB$L_RTT_IRPBL(R5)
        3C A3   D4  079B  1727          CLRL    IRP$L_IOST2(R3)         ; No cancel has been sent yet
                     079E  1728
                     079E  1729 RTT_NETCANSEND:                         ; Send cancel message
                     079E  1730
          019C   30  079E  1731          BSBW    RTT_MAKEIIRP            ; Make iirp for this message
        55 50    E9  07A1  1732          BLBC    R0,RTT_CLEANUP          ; No memory, hangup and goaway
      08D5'CF    9E  07A4  1733          MOVAB   W^RTT_NETWRTDONE,-      ; Place to post io
           0C A2       07A8  1734                  IRP$L_PID(R2)           ;
2C A2    2C A3   D0  07AA  1735          MOVL    IRP$L_SVAPTE(R3), -     ; Move buffer to iirp
                     07AF  1736                  IRP$L_SVAPTE(R2)        ;
           2C A3   D4  07AF  1737          CLRL    IRP$L_SVAPTE(R3)        ; drop it from rtt irp
     51    2C A2   D0  07B2  1738          MOVL    IRP$L_SVAPTE(R2),R1     ; fix the byte count in the iirp
32 A2    0C A1   B0  07B6  1739          MOVW    RBF$W_DATSIZE(R1), -    ; from the size in the buffer
                     07BB  1740                  IRP$W_BCNT(R2)          ;
                     07BB  1741
                     07BB  1742 RTT_NETQUEPKT:                          ; Queue a packet to the netdriver
                     07BB  1743
                     07BB  1744 ;
                     07BB  1745 ;       r2 -> net iirp
                     07BB  1746 ;       r3 -> rtt irp
                     07BB  1747 ;       r5 -> rtt ucb
                     07BB  1748 ;
                     07BB  1749
           38   BB  07BB  1750          PUSHR   #^M<R3,R4,R5>           ; Save the magic three
     53    52   D0  07BD  1751          MOVL    R2,R3                   ; Point to iirp
  55   1C A3   D0  07C0  1752          MOVL    IRP$L_UCB(R3),R5        ; The netucb from this packet
00000000'GF   16  07C4  1753          JSB     G^EXE$ALTQUEPKT         ; Queue iirp to netdriver
           38   BA  07CA  1754          POPR    #^M<R3,R4,R5>           ; restore magic three
     50    01   D0  07CC  1755          MOVL    #1,R0                   ; return success
                05  07CF  1756          RSB
```

07D0 1757

```
                                07D0  1759
                                07D0  1760 ;
                                07D0  1761 ;       R5 -> RTT UCB
                                07D0  1762 ;       R3 -> RTT IRP
                                07D0  1763 ;
                                07D0  1764 ;
                                07D0  1765 ; The net connection is broken, so we must post the irps that come
                                07D0  1766 ; in with an error code.
                                07D0  1767 ;
                                07D0  1768
                                07D0  1769 RTT_NETHUNGUP:
                 50   2C A3      D0  07D0  1770          MOVL    IRP$L_SVAPTE(R3),R0     ; Do we have a buffer
                           0E      13  07D4  1771          BEQL    10$                     ; Nope
                           53      DD  07D6  1772          PUSHL   R3                      ; Push address we care about
                         2C A3     D4  07D8  1773          CLRL    IRP$L_SVAPTE(R3)        ; Forget we had buffer
                    00000000'GF    16  07DB  1774          JSB     G^EXE$DEANONPAGED       ; Get rid of the buffer
                           53  8ED0  07E1  1775          POPL    R3                      ; Restore irp address
          00000000 000020E4 8F     7D  07E4  1776 10$:     MOVQ    #SS$_LINKABORT,-        ; Return a nasty error
                         38 A3        07EE  1777                  IRP$L_IOST1(R3)         ;
                    00000000'GF    16  07F0  1778          JSB     G^COM$POST              ; Post the irp since we don't have
                           50      D4  07F6  1779          CLRL    R0                      ; a link anymore and return error here
                           05      05  07F8  1780          RSB
                                07F9  1781
                                07F9  1782
                                07F9  1783          .SBTTL  RTT_CLEANUP  - Hangup terminal
                                07F9  1784 ;
                                07F9  1785 ; RTT_CLEANUP
                                07F9  1786 ;
                                07F9  1787 ;       We are in deep trouble.  Hangup the terminal to run it down
                                07F9  1788 ;       and return failure in r0.  This is done when we cannot obtain
                                07F9  1789 ;       memory for an iirp or any thing else.  IPL can be anything.
                                07F9  1790 ;
                                07F9  1791 ; inputs:
                                07F9  1792 ;       r5 -> rtt ucb
                                07F9  1793 ;
                                07F9  1794
                                07F9  1795 RTT_CLEANUP:
                                07F9  1796
                    FEF3    30  07F9  1797          BSBW    RTT_HANGUP              ; Post irps and attn asts
                      50    D4  07FC  1798          CLRL    R0                      ; return failure
                            05  07FE  1799          RSB
```

```
                                                                I 3
                     07FF  1801          .SBTTL  RTT_STARTNETRCV  - Start receive to net driver
                     07FF  1802 ;
                     07FF  1803 ; RTT_STARTNETRCV
                     07FF  1804 ;
                     07FF  1805 ;       Start the first receive iirp to the netdriver.  We make an iirp
                     07FF  1806 ;       and queue it to the netdriver with a read function in it.
                     07FF  1807 ;
                     07FF  1808 ; inputs:
                     07FF  1809 ;       r5 -> rtt ucb
                     07FF  1810 ;
                     07FF  1811
                     07FF  1812 RTT_STARTNETRCV:
                     07FF  1813
          00C0 C5    D5  07FF  1814          TSTL    UCB$L_RTT_NETIRP(R5)    ; Is the iirp already out?
                       2E  12  0803  1815          BNEQ    20$                     ; Yes, then ignore it
 00DE C5   0699 8F     B0  0805  1816          MOVW    #SS$_INCOMPAT,UCB$W_RTT_READERR(R5) ; set initial error
                    012E  30  080C  1817          BSBW    RTT_MAKEIIRP            ; Make an iirp for use
                    E7 50  E9  080F  1818          BLBC    R0, RTT_CLEANUP         ; No good, clean it all up
          00C0 C5    52  D0  0812  1819          MOVL    R2, UCB$L_RTT_NETIRP(R5); Save the address of the iirp
       0C A2   0834'CF  9E  0817  1820          MOVAB   W^RTT_NETREADDONE, -    ; Stuff the post address
                       081D  1821                  IRP$L_PID(R2)           ;
             20 A2    21  B0  081D  1822          MOVW    #IO$_READLBLK, -        ; Set the function
                       0821  1823                  IRP$B_FUNC(R2)          ;
                 2C A2  D4  0821  1824          CLRL    IRP$L_SVAPTE(R2)        ; Yes we have no buffer
           00000000'GF  B0  0824  1825          MOVW    G^IOC$GW_MAXBUF,-       ; Set the requested size
                    32 A2       082A  1826                  IRP$W_BCNT(R2)          ;
        00 2A A2    01  E2  082C  1827          BBSS    #IRP$V_FUNC, -          ; Say this is a read function
                       0831  1828                  IRP$W_STS(R2), 10$      ;
                    88  10  0831  1829 10$:     BSBB    RTT_NETQUEPKT           ; and queue the packet to the net
                        05  0833  1830 20$:     RSB
```

```
                                                                .SBTTL  RTT_NETREADDONE  - Post routine for net receive
                                                        ;
                                                        ; RTT_NETREADDONE  Post net receive
                                                        ;
                                                        ;       This is the post routine for receives from the netdriver.
                                                        ;       We look at the packet and send it to the unsolic or interrupt
                                                        ;       routine based on the type of the message.  If the type is
                                                        ;       not recognised or we can't find the irp, we hangup the terminal.
                                                        ;
                                                        ;       We are going to run this code at rtt driver ipl.
                                                        ;
                                                        ; inputs:
                                                        ;       r5 -> net iirp
                                                        ;       ipl = iopost
                                                        ;
                                                        RTT_NETREADDONE:

                38      BB      0834    1850                    PUSHR   #^M<R3,R4,R5>                   ; Save the magic three
                                0836    1851                    DSBINT  #RTT$K_FIPL                     ; Do this work at driver ipl
            53  55      D0      083C    1852                    MOVL    R5,R3                           ; The iirp address is here
        55  10  A3      D0      083F    1853                    MOVL    IRP$L_AST(R3),R5                ; The rtt ucb?
                1C      13      0843    1854                    BEQL    10$                             ; Its gone, we are hung up
          7D 38 A3      E9      0845    1855                    BLBC    IRP$L_IOST1(R3), 60$            ; Error? if so then hang up
        52  2C  A3      D0      0849    1856                    MOVL    IRP$L_SVAPTE(R3), R2            ; The buffer address
            51  62      D0      084D    1857                    MOVL    (R2),R1                         ; Point to message
    50  61  01  A1      0850    1858                    ADDW3   #1,RDP$W_OPCODE(R1),R0          ; Look at the opcode
                14      12      0854    1859                    BNEQ    20$                             ; Its not attention packet
              2C A3     D4      0856    1860                    CLRL    IRP$L_SVAPTE(R3)                ; Buffer not in net packet now
            53  52      D0      0859    1861                    MOVL    R2,R3                           ; Point to buffer with r3
                FDAB    30      085C    1862                    BSBW    RTT_UNSOLIC                     ; Unsolicited input attention message
                39      11      085F    1863                    BRB     40$                             ; Requeue a read
                                0861    1864
                                0861    1865 10$:               ENBINT                                  ; Restore ipl
                38      BA      0864    1866                    POPR    #^M<R3,R4,R5>                   ; Restore all the regs we saved
                006C    30      0866    1867                    BSBW    RTT_NETWRTDONE                  ; Dispose of the iirp and its buffer
                        05      0869    1868                    RSB                                     ;
                                086A    1869
                50      B6      086A    1870 20$:               INCW    R0                              ; Is this an end message?
                58      12      086C    1871                    BNEQ    60$                             ; Nope, hangup the terminal
            50  62      D0      086E    1872                    MOVL    (R2),R0                         ; Point to data
        50  04  A0      D0      0871    1873                    MOVL    RDP$L_REFID(R0),R0              ; Obtain the reference id
                23      13      0875    1874                    BEQL    40$                             ; ** Ignore refids of zero to make
                                0877    1875                                                            ; ** cancel of outofband work
    54  00B8  C5      7E      0877    1876                    MOVAQ   UCB$L_RTT_IRPFL(R5),R4          ; Look through the irps for ours
            51  54      D0      087C    1877                    MOVL    R4,R1                           ; head of queue here
            54  64      D0      087F    1878 30$:               MOVL    (R4),R4                         ; Link through chain
            51  54      D1      0882    1879                    CMPL    R4,R1                           ; end of irps?
                3F      13      0885    1880                    BEQL    60$                             ; Yes, could not find it, hangup
        50  A4  50      D1      0887    1881                    CMPL    R0,IRP$L_SEQNUM(R4)             ; Match? on ref id
                F2      12      088B    1882                    BNEQ    30$                             ; nope
              2C A3     D4      088D    1883                    CLRL    IRP$L_SVAPTE(R3)                ; Buffer not in net iirp now
            53  64      0F      0890    1884                    REMQUE  (R4),R3                         ; Remove the rtt irp from queue
        2C  A3  52      D0      0893    1885                    MOVL    R2,IRP$L_SVAPTE(R3)             ; stick buffer there
                FC37    30      0897    1886                    BSBW    RTT_INTERRUPT                   ; and call interrupt routine
                                089A    1887 40$:
                                089A    1888 ;
```

```
                        089A  1889 ;       16(SP)  RTNADR
                        089A  1890 ;       12(SP)  R5 (iirp address)
                        089A  1891 ;       8(SP)   R4
                        089A  1892 ;       4(SP)   R3
                        089A  1893 ;       0(SP)   SAVED IPL (iopost)
                        089A  1894 ;
 53   0C AE    D0       089A  1895         MOVL    12(SP),R3               ; Obtain the net iirp
 55   1C A3    D0       089E  1896         MOVL    IRP$L_UCB(R3),R5        ; Set the net ucb address up
 50   2C A3    D0       08A2  1897         MOVL    IRP$L_SVAPTE(R3),R0     ; dump the buffer
         0E    13       08A6  1898         BEQL    50$                     ; if there is one to dump
         53    DD       08A8  1899         PUSHL   R3                      ; Save possibly clobbered register
00000000'GF    16       08AA  1900         JSB     G^EXE$DEANONPAGED       ; back into swimming pool
         53  8ED0       08B0  1901         POPL    R3                      ; Restore register
      2C A3    D4       08B3  1902         CLRL    IRP$L_SVAPTE(R3)        ; forget it
00000000'GF    B0       08B6  1903 50$:    MOVW    G^IOC$GW_MAXBUF,-       ; setup for another read from net
      32 A3             08BC  1904         IRP$W_BCNT(R3)          ; with requested buffer size
00000000'GF    16       08BE  1905         JSB     G^EXE$ALTQUEPKT         ; queue to net driver
         09    11       08C4  1906         BRB     70$                     ; Now we are done here
                        08C6  1907
                        08C6  1908 ;
                        08C6  1909 ;       If we had on io error in the packet, then hangup the terminal
                        08C6  1910 ;       deallocate the packet and any buffer and exit.
                        08C6  1911 ;       If there is no rtt ucb left anymore, just deallocate the packet
                        08C6  1912 ;       and buffer and get out.
                        08C6  1913 ;
                        08C6  1914
      FE26    30        08C6  1915 60$:    BSBW    RTT_HANGUP              ; Bad error - hangup the terminal
 55   0C AE    D0       08C9  1916         MOVL    12(SP),R5               ; Net iirp to r5
         06    10       08CD  1917         BSBB    RTT_NETWRTDONE          ; Dump the buffer and the iirp
                        08CF  1918 70$:    ENBINT                          ; Restore the ipl
         38    BA       08D2  1919         POPR    #^M<R3,R4,R5>           ; restore registers of iopost
                  05    08D4  1920         RSB
```

```
                                        .SBTTL  RTT_NETWRTDONE  - Post routine for net write
                                ;
                                ; RTT_NETWRTDONE
                                ;
                                ;       Enter here to post writes to net also.
                                ;       Deallocate the iirp and the message if any.
                                ;
                                ;       r5 -> iirp
                                ;       ipl = iopost or higher
                                ;
                                RTT_NETWRTDONE:

 50    2C A5    D0  08D5  1935          MOVL    IRP$L_SVAPTE(R5),R0     ; Buffer on this iirp?
          02    13  08D9  1936          BEQL    10$                     ; nope
          03    10  08DB  1937          BSBB    20$                     ; deallocate the buffer
    50    55    D0  08DD  1938 10$:     MOVL    R5,R0                   ; Now for the iirp itself
00000000'GF    16  08E0  1939 20$:     JSB     G^EXE$DEANONPAGED       ; back to the pool
               05  08E6  1940          RSB
```

```
                        08E7  1942          .SBTTL  RTT_CANIRPS  - Cancel irps
                        08E7  1943 ;
                        08E7  1944 ; RTT_CANIRPS
                        08E7  1945 ;
                        08E7  1946 ;       Cancel irps by sending a message to the terminal system.
                        08E7  1947 ;
                        08E7  1948 ; inputs:
                        08E7  1949 ;       r4 -> pcb for process
                        08E7  1950 ;       r5 -> rtt ucb
                        08E7  1951 ;       r6 -> channel
                        08E7  1952 ;
                        08E7  1953
                        08E7  1954 RTT_CANIRPS:
                        08E7  1955
        007C 8F    BB   08E7  1956          PUSHR   #^M<R2,R3,R4,R5,R6>
    56  00B8 C5    7E   08EB  1957          MOVAQ   UCB$L_RTT_IRPFL(R5),R6  ; Point to the irp queue
             56    DD   08F0  1958          PUSHL   R6                      ; save its address
                        08F2  1959 ;
                        08F2  1960 ;       20(SP)  R6
                        08F2  1961 ;       16      R5
                        08F2  1962 ;       12      R4
                        08F2  1963 ;       8       R3
                        08F2  1964 ;       4       R2
                        08F2  1965 ;       0       IRP LIST HEAD
                        08F2  1966
        56   66    D0   08F2  1967 10$:     MOVL    (R6),R6                 ; Point to next irp
        6E   56    D1   08F5  1968          CMPL    R6,(SP)                 ; End of queue?
             3E    13   08F8  1969          BEQL    20$                     ; Yes
28 A6     14 AE    B1   08FA  1970          CMPW    20(SP),IRP$W_CHAN(R6)   ; Is this the correct channel?
             F1    12   08FF  1971          BNEQ    10$                     ; Nope, try next?
0C A6     60 A4    D1   0901  1972          CMPL    PCB$L_PID(R4), -        ; Do the pids match?
                        0906  1973                  IRP$L_PID(R6)           ;
             EA    12   0906  1974          BNEQ    10$                     ; Nope, try next
        53   56    D0   0908  1975          MOVL    R6,R3                   ; Set up as the irp of choice
          3C A3    D5   090B  1976          TSTL    IRP$L_IOST2(R3)         ; Did we send a cancel?
             28    12   090E  1977          BNEQ    20$                     ; We are done. just return
        51   18    D0   0910  1978          MOVL    #RBF$W_UNIT+2, R1       ; Get a message buffer for cancel
             53    DD   0913  1979          PUSHL   R3                      ; Save across call
  00000000'GF    16   0915  1980          JSB     G^EXE$ALONONPAGED       ;
             53 8ED0   0918  1981          POPL    R3                      ; Its clobbered if quick irps are gone
           11 50    E9   091E  1982          BLBC    R0,15$                  ; If error, just say we did it
            FB7E    30   0921  1983          BSBW    SET_MSGHDR              ; build the message
                        0924  1984          ASSUME  RBF$W_MOD EQ -
                        0924  1985                  RBF$W_OPCODE+2
             38    D0   0924  1986          MOVL    #IO$_ACPCONTROL,-       ; The message opcode and modifier
          0E A2          0926  1987                  RBF$Q_OPCODE(R2)        ;
             0A    B0   0928  1988          MOVW    #RDP$Q_UNIT+2,-         ; The datasize
          0C A2          092A  1989                  RBF$W_DATSIZE(R2)       ;
                        092C  1990 ;       MOVL    R2,IRP$L_SVAPTE(R3)     ; Save the buffer address **
            FE6F    30   092C  1991          BSBW    RTT_NETCANSEND          ; Send the message
          06 50    E9   092F  1992          BLBC    R0,20$                  ; Error, IRPS are all gone
3C A3        01    D0   0932  1993 15$:     MOVL    #1,IRP$L_IOST2(R3)      ; Mark for we sent it
             BA    11   0936  1994          BRB     10$                     ; try another irp
                        0938  1995
        007E 8F    BA   0938  1996 20$:     POPR    #^M<R1,R2,R3,R4,R5,R6>  ; Restore regs and return
                   05   093C  1997          RSB                             ; Discard stack longword to r1
```

```
                        093D  1999          .SBTTL RTT_MAKEIIRP  - Manufacture an internal irp
                        093D  2000 ;
                        093D  2001 ; RTT_MAKEIIRP
                        093D  2002 ;
                        093D  2003 ;       Make an internal IRP for sending to the netdriver.
                        093D  2004 ;       If we can't get the space, return failure.
                        093D  2005 ;
                        093D  2006 ; inputs:
                        093D  2007 ;       r3 -> rtt irp
                        093D  2008 ;       r5 -> rtt ucb
                        093D  2009 ;
                        093D  2010 ; outputs:
                        093D  2011 ;       r0 = success or fail
                        093D  2012 ;
                        093D  2013
                        093D  2014 RTT_MAKEIIRP:
                        093D  2015
 51    C4 8F    9A      093D  2016          MOVZBL  #IRP$C_LENGTH,R1        ; Obtain a buffer of correct size
          53    DD      0941  2017          PUSHL   R3                      ; Save across call to get memory
00000000'GF    16      0943  2018          JSB     G^EXE$ALONONPAGED       ; from dynamic memory
          53 8ED0      0949  2019          POPL    R3                      ; Restore irp address
       3A 50    E9      094C  2020          BLBC    R0,10$                  ; No memory left, so return error
 0A A2    0A    90      094F  2021          MOVB    #DYN$C_IRP, -           ; Set the type and size fields
                        0953  2022                  IRP$B_TYPE(R2)          ;
 08 A2    51    B0      0953  2023          MOVW    R1,IRP$W_SIZE(R2)       ;
    0C A2       D4      0957  2024          CLRL    IRP$L_PID(R2)           ; No pid here
 10 A2    55    D0      095A  2025          MOVL    R5,IRP$L_AST(R2)        ; Save the rtt ucb field
    00B4 C5    D0      095E  2026          MOVL    UCB$L_RTT_NETWIND(R5),- ; Set up the window
       18 A2            0962  2027                  IRP$L_WIND(R2)          ;
    00B0 C5    D0      0964  2028          MOVL    UCB$L_RTT_NETUCB(R5),-  ; and the ucb for the net
       1C A2            0968  2029                  IRP$L_UCB(R2)           ;
          20    B0      096A  2030          MOVW    #IO$_WRITELBLK,-        ; the function
       20 A2            096C  2031                  IRP$W_FUNC(R2)          ;
23 A2    04    90      096E  2032          MOVB    #4,IRP$B_PRI(R2)        ; priority of this in queue
          01    B0      0972  2033          MOVW    #IRP$M_BUFIO,-          ; Its a buffered io function
       2A A2            0974  2034                  IRP$W_STS(R2)           ; and assume a write
       30 A2    B4      0976  2035          CLRW    IRP$W_BOFF(R2)          ; no quota to return for iirp
       38 A2    7C      0979  2036          CLRQ    IRP$L_IOST1(R2)         ; no status yet
                        097C  2037          ASSUME  IRP$L_OBCNT -
                        097C  2038          EQ -
                        097C  2039                  IRP$L_ABCNT+4
       40 A2    7C      097C  2040          CLRQ    IRP$L_ABCNT(R2)         ; Some more byte counts
       50 A3    D0      097F  2041          MOVL    IRP$L_SEQNUM(R3),-      ; Grab a quick sequence number
       50 A2            0982  2042                  IRP$L_SEQNUM(R2)        ;
       58 A3    D0      0984  2043          MOVL    IRP$L_ARB(R3),-         ; Access rights block, incase needed
       58 A2            0987  2044                  IRP$L_ARB(R2)           ;
                05      0989  2045 10$:     RSB
```

```
                        .SBTTL  RTT_END, End of driver

                ;
                ; Label that marks the end of the driver
                ;
RTT_END:
                        .END
```

```
$$$                     =  00000020  R     02
$$OP                    =  00000002
ABORT                      000003B9  R     03
ACB$L_KAST              =  00000018
ALLOC_ABORT                00000478  R     03
ALLOC_MESSAGE              0000047E  R     03
AQB$L_ACPPID            =  0000000C
AT$_NULL                =  00000005
BUFADDR                 =  00000000
BUFSIZE                 =  00000004
BUG$_BRDMSGLOST            ********    X   03
CAN$C_CANCEL            =  00000000
CAN$C_DASSGN            =  00000001
CHK_READERR                00000276  R     03
COM$DELATTNAST             ********    X   03
COM$DELCTRLAST             ********    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
CTRLC                      0000066A  R     03
CTRLY                      00000671  R     03
CTRL_CY                    00000347  R     03
DC$_TERM                =  00000042
DDB$L_ACPD              =  00000010
DDB$L_DDT               =  0000000C
DDB$T_NAME              =  00000014
DELAST                     00000676  R     03
DEV$M_AVL               =  00040000
DEV$M_CCL               =  00000002
DEV$M_IDV               =  04000000
DEV$M_NNM               =  00000200
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$REINITAB               00000081  R     02
DPT$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$C_UCB               =  00000010
EXE$ABORTIO                ********    X   03
EXE$ALLOCBUF               ********    X   03
EXE$ALONONPAGED            ********    X   03
EXE$ALTQUEPKT              ********    X   03
EXE$BUFFRQUOTA             ********    X   03
EXE$DEANONPAGED            ********    X   03
EXE$FINISHIOC              ********    X   03
EXE$INSERTIRP              ********    X   03
EXE$MAXACMODE              ********    X   03
EXE$PROBER                 ********    X   03
EXE$QIORETURN              ********    X   03
EXE$READCHK                ********    X   03
EXE$SNDEVMSG               ********    X   03
EXE$WRITECHK               ********    X   03
EXE$WRTMAILBOX             ********    X   03
FDT_FINISHIOC              000003C2  R     03
FDT_FINISHIOC_OK           000003BF  R     03
FUNCTAB_LEN             =  00000040
GET_PARAMS                 000003C8  R     03
HANGUP                     00000660  R     03
INIADDR                 =  00000018
INIOFFSET               =  00000024
INISIZE                 =  0000001C
IO$M_CTRLCAST           =  00000100
IO$M_CTRLYAST           =  00000080
IO$M_EXTEND             =  00008000
IO$M_HANGUP             =  00000200
IO$M_OUTBAND            =  00000400
IO$M_TIMED              =  00000080
IO$M_TYPEAHDCNT         =  00000040
IO$V_BRDCST             =  0000000E
IO$V_BREAKTHRU          =  00000009
IO$V_EXTEND             =  0000000F
IO$V_INCLUDE            =  0000000B
IO$V_MAINT              =  00000006
IO$V_RD_MODEM           =  00000007
IO$_ACPCONTROL          =  00000038
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$RETURN                 ********    X   03
IRP$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_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$Q_TT_STATE          =  00000040
IRP$S_FCODE             =  00000006
IRP$V_FCODE             =  00000000
IRP$V_FUNC              =  00000001
IRP$W_BCNT              =  00000032
IRP$W_BOFF              =  00000030
IRP$W_CHAN              =  00000028
IRP$W_FUNC              =  00000020
IRP$W_RTT_COMPAT        =  00000040
IRP$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
POST                       00000524 R      03
POST_BROADCAST             00000545 R      03
POST_SENSE                 00000506 R      03
PR$_IPL                 =  00000012
PRMADDR                 =  00000008
PRMSIZE                 =  0000000C
RBF$B_TT_OUTBAND        =  00000018
RBF$B_TYPE              =  0000000A
RBF$C_TT_UNSOL          =  00000000
RBF$K_HEADERLEN         =  00000018
RBF$L_MSGDAT            =  00000000
RBF$L_PARAM1            =  00000018
RBF$L_REFID             =  00000012
RBF$L_TT_BCNT           =  00000018
RBF$L_TT_CARCON         =  0000001C
RBF$L_TT_CHAR2          =  0000002C
RBF$L_TT_FILL           =  00000024
RBF$L_TT_PARITY         =  00000028
RBF$L_TT_SPEED          =  00000020
RBF$L_TT_TIMOUT         =  0000001C
RBF$L_USRBFR            =  00000004
RBF$Q_TT_CHAR           =  00000018
RBF$T_TT_TERM           =  00000020
RBF$T_TT_WDATA          =  00000020
RBF$W_DATSIZE           =  0000000C
RBF$W_MOD               =  00000010
RBF$W_OPCODE            =  0000000E
RBF$W_SIZE              =  00000008
RBF$W_UNIT              =  00000016
RDP$B_TT_OUTBAND        =  0000000A
RDP$C_TT_BRDNAME        =  00000010
RDP$L_REFID             =  00000004
RDP$L_TT_SCHAR2         =  0000001A
RDP$Q_STATUS            =  0000000A
RDP$Q_TT_SCHAR          =  00000012
RDP$T_TT_BRDNAME        =  00000010
RDP$T_TT_RDATA          =  00000012
RDP$W_MOD               =  00000002
RDP$W_OPCODE            =  00000000
RDP$W_TT_BRDMSG         =  0000000C
RDP$W_TT_BRDTOTSIZE     =  0000000A
RDP$W_TT_BRDUNIT        =  0000000E
RDP$W_UNIT              =  00000008
READ_ERROR                 0000019A R      03
READ_LOCAL              =  00000028
REM$C_CURECO            =  00000001
REM$C_CURVRS            =  00000001
REM$C_LNK_READ          =  00000002
REM$C_MAXDEVS           =  0000000A
REM$C_MAXLINKS          =  00000010
REM$C_MAXUNITS          =  00000010
REM$C_MBX_READ          =  00000001
REM$C_ST_ATTRIB         =  00000002
REM$C_ST_CONFIG         =  00000001
RTT$DDT                    00000000 RG     03
RTT$K_FIPL              =  00000008
RTT_ABORTIRPS              0000071D R      03
RTT_BRDCST                 0000068D R      03
RTT_CANCEL                 00000564 R      03
RTT_CANIRPS                000008E7 R      03
RTT_CHARSIZE               000003D9 R      03
RTT_CLEANUP                000007F9 R      03
RTT_ECOQ                   000003F8 R      03
RTT_END                    0000098A R      03
RTT_FUNCTABLE              00000038 R      03
RTT_HANGUP                 000006EF R      03
RTT_INTERRUPT              000004D1 R      03
RTT_MAKEIIRP               0000093D R      03
RTT_NETCANSEND             0000079E R      03
RTT_NETHUNGUP              000007D0 R      03
RTT_NETMSGSEND             00000783 R      03
RTT_NETMSGSENDX            0000077B R      03
RTT_NETQUEPKT              000007BB R      03
RTT_NETREADDONE            00000834 R      03
RTT_NETWRTDONE             000008D5 R      03
RTT_OUTBAND                000006CD R      03
```

```
RTT_READ                        000000C5 R      03
RTT_SENSEMODE                   00000408 R      03
RTT_SETMODE                     00000287 R      03
RTT_STARTNETRCV                 000007FF R      03
RTT_UNSOLIC                     0000060A R      03
RTT_WRITE                       00000078 R      03
RT_READ_ITMLST                  000001A0 R      03
SCH$WAKE                        ******** X      03
SENSE_SPAWN                     0000054A R      03
SET_BRDCST                      00000303 R      03
SET_CHAR                        000002C8 R      03
SET_CONNECT                     000002FB R      03
SET_CTRLC                       0000033C R      03
SET_CTRLY                       00000316 R      03
SET_DISCONNECT                  000002FB R      03
SET_HANGUP                      0000034A R      03
SET_MAINT                       000002FB R      03
SET_MESSAGE                     0000034A R      03
SET_MSGHDR                      000004A2 R      03
SET_MSGHDRX                     000004BD R      03
SET_NOP                         00000313 R      03
SET_OUTBAND                     0000037B R      03
SET_PID                         0000030D R      03
SS$_ABORT                   =   0000002C
SS$_ACCVIO                  =   0000000C
SS$_BADPARAM                =   00000014
SS$_DEVREQERR               =   00000334
SS$_HANGUP                  =   000002CC
SS$_ILLIOFUNC               =   000000F4
SS$_INCOMPAT                =   00000699
SS$_LINKABORT               =   000020E4
SS$_NORMAL                  =   00000001
STARTRCV                        00000665 R      03
TIMEOUT                     =   00000020
TRM$_LASTITM                =   0000000A
TRMADDR                     =   00000010
TRMSIZE                     =   00000014
TT$V_HALFDUP                =   00000014
TT$_UNKNOWN                 =   00000000
TT2$M_DCL_MAILBX            =   00000200
TT2$V_BRDCSTMBX             =   00000004
TTY$GL_DEFCHAR                  ******** X      02
TTY$GL_JOBCTLMB                 ******** X      03
TTY$GL_OWNUIC                   ******** X      02
TTY$GW_DEFBUF                   ******** X      02
TTY$GW_PROT                     ******** X      02
UCB$B_DEVCLASS              =   00000040
UCB$B_DEVTYPE               =   00000041
UCB$B_DIPL                  =   0000005E
UCB$B_FIPL                  =   0000000B
UCB$B_RTT_PRDECO            =   000000D5
UCB$K_RTT_LEN               =   00000138
UCB$K_RTT_LENGTH            =   00000138
UCB$L_AMB                   =   00000060
UCB$L_DDB                   =   00000028
UCB$L_DEVCHAR               =   00000038
UCB$L_DEVCHAR2              =   0000003C
UCB$L_DEVDEPEND             =   00000044
UCB$L_DEVDEPND2             =   00000048
UCB$L_RTT_BANDEXCL          =   0000009C
UCB$L_RTT_BANDEXMSK         =   00000098
UCB$L_RTT_BANDINCL          =   000000C4
UCB$L_RTT_BANDINMSK         =   000000C8
UCB$L_RTT_CTRLC             =   00000094
UCB$L_RTT_CTRLY             =   00000090
UCB$L_RTT_DEVDEPEND2        =   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_SVAPTE                =   00000078
UCB$L_SVPN                  =   00000074
UCB$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$M_TT_HANGUP             =   00000008
UCB$Q_TL_BRKTHRU            =   000000A8
UCB$V_JOB                   =   00000000
UCB$V_ONLINE                =   00000004
UCB$V_TT_HANGUP             =   00000003
UCB$W_CT_QCTPCNT            =   000000DE
UCB$W_DEVBUFSIZ             =   00000042
UCB$W_DEVSTS                =   00000068
UCB$W_REFC                  =   0000005C
UCB$W_RTT_READERR           =   000000DE
UCB$W_STS                   =   00000064
UCB$W_UNIT                  =   00000054
UNSOLIC_EXIT                    0000067C R      03
UNSOL_DATA                      0000062C R      03
VCB$L_AQB                   =   00000010
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$105_PROLOGUE	0000008C (140.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	0000098A (2442.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.05	00:00:01.45
Command processing	138	00:00:00.48	00:00:03.43
Pass 1	801	00:00:25.20	00:01:30.56
Symbol table sort	0	00:00:03.86	00:00:13.19
Pass 2	351	00:00:05.52	00:00:21.78
Symbol table output	38	00:00:00.22	00:00:00.37
Psect synopsis output	3	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1365	00:00:35.34	00:02:10.80

The working set limit was 2700 pages.
211039 bytes (413 pages) of virtual memory were used to buffer the intermediate code.
There were 190 pages of symbol table space allocated to hold 3595 non-local and 92 local symbols.
2053 source lines were read in Pass 1, producing 23 object records in Pass 2.
62 pages of virtual memory were used to define 59 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]REM.MLB;1	2
_255DUA28:[SYS.OBJ]LIB.MLB;1	39
_255DUA28:[SYSLIB]STARLET.MLB;2	15
TOTALS (all libraries)	56

3925 GETS were required to define 56 macros.

There were no errors, warnings or information messages.

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

PUDRIVER
LIS

RTTDRIVER
LIS

PATABLES
LIS

PASCSCTL
LIS