NBSIR 84-2902

Selected NBSNET Software

Michael Strawbridge
Sheryl Schooley
Joseph Sokol, Jr.*
Robert Crosson

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Applied Mathematics
Gaithersburg, MD 20899

*Presently with Interactive Systems Corporation
Gaithersburg, MD

September 1984

S. DEPARTMENT OF COMMERCE
TIONAL BUREAU OF STANDARDS

NBSIR 84-2902

SELECTED NBSNET SOFTWARE

Michael Strawbridge
Sheryl Schooley
Joseph Sokol, Jr.*
Robert Crosson

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Applied Mathematics
Gaithersburg, MD 20899

*Presently with Interactive Systems Corporation
Gaithersburg, MD

September 1984

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

Selected NBSNET Software

February 1984

by

Michael Strawbridge
Sheryl Schooley
Joseph Sokol, Jr.*
Robert Crosson

Abstract

NBSNET is a local area communications network at the National Bureau of Standards. Ethernet-like in its design, it has operated successfully since 1979, supporting terminal-computer and computer-computer communications. Devices physically connect to NBSNET through RS-232-C interfaces; each being customized to the device being served. Over 600 physical connections currently are in use. Customization primarily involves modifying the control program, called a "personality", for each interface. Each personality is divided into modules which implement, among other things, the network's internal protocol and the external device communications protocol. Three external device protocols are used. A listing of the network protocol software and some typical personality modules is supplied.

* Presently with Interactive Systems Corporation, Gaithersburg, Maryland.

Selected NBSNET Software

February, 1984

by

Michael Strawbridge
Sheryl Schooley
Joseph Sokol, Jr.*
Robert Crosson

Introduction

HISTORY. The National Bureau of Standards is a Federal Government research facility with branches in Gaithersburg, Maryland, and Boulder, Colorado. In 1976 the Institute for Computer Sciences and Technology began designing for NBS a local area communications network called NBSNET. It became operational in 1979 and is now supported by the Network Support Group of the Scientific Computing Division. The system is similar to Ethernet in that it transfers data packets over coaxial cables using the Carrier-Sense Multiple-Access medium access method with Collision Detection (CSMA/CD). The tranmission rate over the coaxial cables, however, is one megabit/second.

CABLE TOPOLOGY. In Gaithersburg, NBSNET covers many acres and serves 18 buildings. In Boulder, six buildings are served. The two local segments, together consisting of over 600 nodes, are connected by a leased telephone line operating at 9600 bits per second. Another telephone link extends from the Boulder facility to Colorado State University at Ft. Collins. An optical link connects the Joint Institute for Laboratory Astrophysics, University of Colorado, with the Boulder branch of NBSNET.

Hardware Configuration

USER BOARDS. Users connect to the network through RS-232-C interfaces called User Boards. One User Board is required per physical connection. User Boards contain the network operation software, which uses the internal network protocol to direct the communication between two User Boards. The software also assembles network packets with the data received from the user, and disassembles network data packets and passes the data to the user. User control of the network software is accomplished by command messages.

* Presently with Interactive Systems Corporation, Gaithersburg, Maryland.

NETWORK BOARDS AND TAPS. A Network Board acts as an interface between the TAP and up to eight User Boards. When a User Board has a packet for transmission, it notifies the Network Board. The Network Board determines if the network is idle, and if so, begins encoding data and sending it to the TAP. The TAP provides an electrical connection to the coaxial cable without disturbing its transmission characteristics.

CUSTOMIZATION. Each User Board in NBSNET is customized for the device to which it is connected. Customization is accomplished through hardware strapping and the creation of a "personality" by modification of the User Board's software. The existence of personalities permits a more complete adaptation of the network interface to the user's equipment. This allows equipment to operate at a level closer to its full potential than could be accomplished with a standard interface.

Software Functions

PERSONALITIES. Three general types of personalities are available. One is for terminals; two are for computers. The personalities in general provide programmable options such as the ability to map characters into other, predefined characters, and the ability to control the flow of data between the user and the network when either side is temporarily unable to accept it.

TERMINALS. Terminal personalities operate in two modes; normal mode and command mode. In normal mode, the User Board accepts characters as information to be transmitted over the network. Command mode enables the user to communicate with the User Board to make or break connections, to determine the status of his connection, or to alter through the User Board's software the personality of his network node.

COMPUTERS. Computer personalities are divided into terminal emulation and data integrity checking types. Terminal emulation computer personalities implement a simple link level protocol which passes characters between the computer and the User Board. A program is required in the computer to interact with the user to obtain the necessary information required by the computer to perform network transactions.

A data integrity checking personality provides functions beyond the terminal emulation type through the implementation of a more specialized link protocol. This protocol affects communication only between the computer and the User Board, and uses a modified HDLC protocol for error detection. Since internal network error dectection is guaranteed, the addition of this personality provides end-to-end error detection, along with the expected facilities for circuit establishment and maintenance. A program is required in the computer to perform the necessary interactive functions for the data integrity checking personality to operate properly.

Software Organization

MODULES. The structure of the User Board software reflects the major responsibilities of interfacing devices to NBSNET. The software is divided into five major modules:

a) User Read
b) User Write
c) Network Read
d) Network Write
e) Scheduler

SCHEDULER. The User Board software operates without interrupts. The various tasks to be performed are scheduled to be run at appropriate times by the Scheduler module. Modules are scheduled in round-robin fashion and are only scheduled to run when necessary. Read modules are always scheduled since there are no interrupts and all incoming data must be read. Write modules are scheduled only when data is available to send. Also, various other support functions (such as timer control) are performed in the scheduler.

NETWORK MODULES. The network side of the software (Net Read and Net Write) performs the various protocol functions to establish, control, and maintain virtual circuits on the network. This includes establishment of connections, sending all packets, sequencing packet numbers, verifying acknowledgements, etc. Data buffers and various control functions (such as interrupt and flow control) are communicated to the user side in this module.

In the past, some of the interactions with the user side of the software have been personalized for the user device. This practice has been halted and a standard interface is being created between the network and user sides.

USER MODULES. The user side of the software (User Read and User Write) sends data to and receives data from the user's device, and communicates the various control functions such as interrupt, connection established, connection broken, etc. Since these modules are tailored for specific devices, they are the portion of the software which distinguishes one personality from another. For instance, a terminal personality will normally send and receive data asynchronously, with the control messages between the User Board and the user being in human readable form. But with a data integrity checking personality, the data being transferred between the computer and its User Board are enclosed in an HDLC I-frame, with the control messages being part of either the HDLC or network level protocols.

Listings

The following listings represent the modules assembled to produce a simple User Board personality. The language used is a variation of 6502 microprocessor assembly language, and is available from NBS. Not all of the modules available for generating personalities are represented. These modules give an indication of what is involved in producing a personality. The listings are subject to change because the personality software is continually undergoing revision. For further information contact the NBSNET Support Group, Scientific Computing Division, Center for Applied Mathematics, National Engineering Laboratory, National Bureau of Standards, Gaithersburg, MD 20899.

Note

Much of the basic design of the User Board software was the work of Brian G. Lucas.

Additional Information

Paul D. Amer, Robert Rosenthal, and Robert Toense; "Measuring a Local Network's Performance", Data Communications, April, 1983.

R. Carpenter, J. Sokol, and R. Rosenthal; "A Microprocessor-based Local Network Node"; Proceedings, COMPCON '78 Fall; IEEE Computer Society.

Robert J. Carpenter, J. Eryx Malcolm, and Michael L. Strawbridge; "Operational Experience with the NBS Local Area Network"; Local Networks for Computer Communications; A. West, P. Janson (eds.); North-Holland Publishing Company, 1981.

Robert J. Crosson; "Operating a Local Area Network"; Proceedings, Computer Networking Symposium, December, 1983; IEEE Computer Society.

Daniel P. Stokesberry; "A Characterization of Traffic on NBSNET"; Workshop on Performance and Evaluation of Local Area Networks; Worcester Polytechnic Institute, 1983.

Robert E. Toense; "Performance Analysis of NBSNET"; Workshop on Performance and Evaluation of Local Area Networks; Worcester Polytechnic Institute, 1983.

FEDERAL INFORMATION PROCESSING STANDARD SOFTWARE SUMMARY

01. Summary date
Yr. 84 Mo. 06 Day 14

02. Summary prepared by *(Name and Phone)*
Robert Crosson, (301) 921-2562

03. Summary action
New ☒ Replacement □ Deletion □
Previous Internal Software ID

04. Software date
Yr. 84 Mo. 02 Day 01

05. Software title
Selected NBSNET Software - February 1984

06. Short title

07. Internal Software ID
netsoft #1

08. Software type
□ Automated Data System
☒ Computer Program
□ Subroutine/Module

09. Processing mode
☒ Interactive
□ Batch
□ Combination

10. Application area

General
□ Computer Systems Support/Utility
□ Scientific/Engineering
□ Bibliographic/Textual
□ Management/Business
□ Process Control
☒ Other

Specific
Local area network control program

11. Submitting organization and address

Network Support Group
Scientific Computing Division
National Bureau of Standards
Gaithersburg, MD 20899

12. Technical contact(s) and phone

Sheryl Schooley (301) 921-2145
Robert Crosson (301) 921-2562

13. Narrative

This program controls the operation of a local area network (NBSNET) node. It is a stand-alone program running on a 6502 microprocessor, which controls virtual circuit management over the network.

14. Keywords

communications;computer;LAN;NBSNET;network;protocol

15. Computer manuf'r and model
Mostek 6502 micropro.

16. Computer operating system
none

17. Programing language(s)
6502 Assembly Lang.

18. Number of source program statements
4874

19. Computer memory requirements
4096 bytes

20. Tape drives
none

21. Disk/Drum units
none

22. Terminals
one

23. Other operational requirements
NBSNET Terminal Interface Equipment

24. Software availability
Available ☒ Limited □ In-house only □

Contact S. Schooley (301) 921-2145 or
R. Crosson (301) 921-2562. Form: listings,
UNIX tar format 1600 bpi mag-tape, Read-
only Memory (2716's).

25. Documentation availability
Available □ Inadequate ☒ In-house only □

26. FOR SUBMITTING ORGANIZATION USE

```
1000          ai0data = $1000         ;address of 2651-0 data reg
1001          ai0stat = $1001         ;address of 2651-0 status reg
1002          ai0mode = $1002         ;address of 2651-0 mode reg
1003          ai0cntr = $1003         ;address of 2651-0 control reg

0001          trdy = 1                ;transmitter buffer is empty
0002          rrdy = 2                ;reciever buffer is full
0020          FRerr= $20              ;framing error
0040          DCD  = $40              ;data carrier detect
0080          RTS  = $80              ;RTS detect

0000          NUL= $00
0007          BEL= $07
0008          BS = $08
0009          HT = $09
000A          LF = $0A
000B          VT = $0B
000C          FF = $0C
000D          CR = $0D
001A          SUB= $1A
0005          ENK= $05
0004          EOT= $04
0020          SP = $20

0003          CNTLC = $03
0004          CNTLD = $04
0011          CNTLQ = $11
0012          CNTLR = $12
0013          CNTLS = $13
0014          CNTLT = $14
0015          CNTLU = $15
0017          CNTLW = $17

001B          ESC = $1B

01E0          totermsp = $01E0        ;location of initial stack for process that writes term
01C0          fmnetsp = $01C0         ;location of initial stack for process that reads net
01A0          tonetsp = $01A0         ;location of initial stack for process that writes net

FF00          DEFTAB = $FF00          ;start of default terminal parameters
C000          orb6522 = $C000         ;location of output register b
C001          ora6522 = $C001         ;location of output register a

C002          ddrb6522 = $C002        ;location of data direction for port b
C003          ddra6522 = $C003        ;location of data direction for port a

C004          t1l6522 = $C004         ;location of timer 1 low order count
C005          t1h6522 = $C005         ;location of timer 1 high order count (load last)
C008          t2l6522 = $C008         ;location of timer 2 low order count
C009          t2h6522 = $C009         ;location of timer 2 high order count (load last)

C00B          acr6522 = $C00B         ;location of auxiliary control register
C00C          pcr6522 = $C00C         ;location of peripheral control register
C00D          ifr6522 = $C00D         ;location of interrupt flag register
C00E          ier6522 = $C00E         ;location of interrupt enable register
```

```
1800      TIEaddrh = $1800        ;address of register which contains high half of TIE's address
1801      TIEaddrl = $1801        ;address of register which contains the low half


1C00      HARDWD = $1C00          ;address of register to tickle to reset hardware watchdog

0100      rptaddr = $0100         ;net address of statistics logger

0000      PAGE0 = $0000           ;address of page where most of the state variables are kept
0200      PAGE2 = $0200           ;the rest of them are on page 2

          ;QBFqbfpckt = $0390             ;address of qbf packet
          ;QBF
0020      t2f     = $20           ;indicates clock has ticked

0040      success = $40           ;indicates a successful transmission of a packet

1C02      clrrb0 = $1C02          ;address to tickle to release receive buffer 0
1C04      clrrb1 = $1C04          ;address to tickle to release receive buffer 1
1C06      clrrb2 = $1C06          ;address to tickle to release receive buffer 2

0500      rcv0buf = $0500         ;address of receive buffer 0
0600      rcv1buf = $0600         ;address of receive buffer 1
0700      rcv2buf = $0700         ;address of receive buffer 2

00AA      trbuf0pt = $AA          ;pointer to transmit buffer 0
00AB      trbuf1pt = $AB          ;pointer to transmit buffer 1
00A4      trbuf2pt = $A4          ;pointer to transmit buffer 2

0005      HEADSIZ = 5             ;size of header for normal packet
                                  ;may be 1 larger if type is ESCAPE
002A      RPTSIZ = 42             ;size of report datagram

0000      hardcnt = 0             ;position of char count in transmit buffer
0001      dstaddr = 1             ;position of destination address in transmit buffer
0003      srcaddr = 3             ;position of source address in transmit buffer
0005      packtyp = 5             ;position of packet type in transmit buffer
0006      trstart = 6             ;position of data field in transmit buffer

0200      trbuf0 = $0200          ;address of transmit buffer 0 (data)
0300      trbuf1 = $0300          ;address of transmit buffer 1 (data)
0400      trbuf2 = $0400          ;address of transmit buffer 2 (control)


0430      echo.start = $0430      ;buffer filled from keyboard or internal process
                                  ;emptied to screen

0001      BREAK = 1               ;if low order bit is set, char causes line to be
                                  ;transmitted

0001      BUSY = 1                ;TIE is either connected or in process of connecting
                                  ;other connection requests will now be ignored

0001      SDISC = 1               ;sent a DISCON packet
0002      RDISC = 2               ;received a DISCON packet
                                  ;once both of these events have occurred, the TIE is
```

```
                                            ;considered to be disconnected

              0008      RRb = $08           ;flow control bit; enables the local TIE to send data
                                            ;it is set by the distant TIE

                                            ;network packet types

              0000      NOP     = 0
              0001      DATA    = 1
              0004      CONN    = 4
              0005      DISCON  = 5
              0007      ESCAPE  = 7
              0080      INTR1   = $80
              0081      INTR2   = $81
              0082      ENQ     = $82
              0083      STTY    = $83
              0084      GTTY    = $84
              0085      CTTY    = $85
              0086      SSERV   = $86
              0087      REPT    = $87


                                            ;network states
              0000      IDLE    = $00       ;TIE is in idle state (able to establish a connection)
              0008      SCON    = $08       ;TIE has just sent a request for connection
              0010      RCON    = $10       ;TIE has just received a request for connection
              0018      CONNECT = $18       ;TIE is in the connected state

              000F      COLMAX  = 15        ;maximum number of collisions per packet

              0002      PROCMAX = 2         ;maximum index into table of process to run

              0009      PARM.MAX = 9        ;number of paramaters in paramter message

              0000      fcnop = 0           ;no function for this char
              0002      fcdlw = 2           ;back up cursor over last word, erase rest of line
              0004      fcdlc = 4           ;back up cursor one postion, and erase rest of line
              0006      fcdll = 6           ;back up cursor to last prompt, erase rest of line
              0008      fcrpl = 8           ;print current input text out again
              000A      fctnl = 10          ;map character into new-line
              000C      fcstp = 12          ;stop output to the screen
              000E      fcstr = 14          ;start output to the screen
              0010      fcint = 16          ;generate interrupt to host, flush buffers
              0012      fcesc = 18          ;pass through next char unmolested
              0014      fctie = 20          ;set to tie command mode



              0000              org     PAGE0


     0000     0002      hardwd: rmb     2   ;contains address of hardware watchdog register
                                            ;used "indirectly" to help insure integrity of
                                            ;the data on page 0     (must be at location 0)

     0002     0001      initall:rmb     1   ;when 1 indicates all pararamters (including terminal)
```

```
                                         ;are to be initialized. if 0, the terminal paramaters
                                         ;such as the break class chars, do not get changed

                 ;QBFdiag:       rmb     1       ;when 1, indicates TIE in diag mode
0003     0001    curproc:rmb     1       ;current process number stored here


                 ;this table contains the status of each of the processes to be scheduled
                 ;a zero entry means do not schedule this process at this time
                 ;a non-zero entry means go ahead and run this process

0004             jobstat:
0004     0001    outterm: rmb    1       ;status of write term process
0005     0001    netread: rmb    1       ;status of read net process
0006     0001    netwrite:rmb    1       ;status of write net process


                 ;this table contains the address of the current stack for each of
                 ;the processes that are to be scheduled

0007     0001    jobsp:  rmb     1       ;stack address of write term process
0008     0001            rmb     1       ;stack address of read net process
0009     0001            rmb     1       ;stack address of write net process


000A     0001    tran.used:rmb   1       ;number of full transmit buffers

000B     0002    tin.buf:rmb     2       ;addr of current transmit buffer being filled
000D     0001    tin.p:  rmb     1       ;input pointer into trans buffer being filled
000E     0001    tin.cnt:rmb     1       ;count of chars in current trans buffer
000F     0001    tin.brk:rmb     1       ;if 1, release current transmit buffer to
                                         ;network tranmsit process

0010     0002    tout.buf:rmb    2       ;addr of current buffer to be transmitted

0012     0001    echo.inp:rmb    1       ;input pointer into echo buffer for term
0013     0001    echo.outp: rmb  1       ;output pointer into echo buffer
0014     0001    echo.free: rmb  1       ;number of unused locations in echo buffer
0015     0001    echo.used: rmb  1       ;number of used locations in echo buffer

0016     0001    parm1:  rmb     1       ;optional first paramater of escape sequence
0017     0001    parm2:  rmb     1       ;optional second paramter of escape sequence
0018     0001    parm3:  rmb     1       ;optional third paramter of escape sequence
0019     0001    parm4:  rmb     1       ;optional fourth paramter of escape sequence
001A     0001    parm5:  rmb     1       ;optional fifth paramter of escape sequence
001B     0001    parm6:  rmb     1       ;optional sixth paramter of escape sequence
001C     0001    parm7:  rmb     1       ;optional seventh paramter of escape sequence
001D     0001    parm8:  rmb     1       ;optional eighth paramter of escape sequence
                                         ;used in cursor positioning

001E     0001    lcol:   rmb     1       ;column loc of last char ouput to term

001F     0001    quecnt: rmb     1       ;count of things in que for net-trans to do
0020     0008    que:    rmb     8       ;reserve some space for the que
0028     0001    queinp: rmb     1       ;input pointer into que
0029     0001    queoutp:rmb     1       ;points at next thing to do
```

```

002A     0001        recinp: rmb      1        ;input pointer into receiver buffer que
002B     0001        recoutp:rmb      1        ;points at next buffer to empty
002C     0001        reccnt: rmb      1        ;current count of bufs to be emptied
002D     0008        recque: rmb      8        ;save some space for reciever buffer que
                                               ;this que contains the addresses of buffers
                                               ;which contain packets received from the net

0035     0001        curbufs:rmb      1        ;bit map of receive bufs not yet released
0036     0001        currbufl:rmb     1        ;used by read-net process to store real
0037     0001        currbufh:rmb     1        ;address of current buffer being emptied

0038     0001        relbufl:rmb      1        ;address of receive buffer
0039     0001        relbufh:rmb      1        ;to be released

003A     0001        fctask: rmb      1        ;when 1, indicates turn flow control on
                                               ;when 2, indicates turn flow control off
003B     0001        fcstat: rmb      1        ;status of flow from device to TIE ... 1 = on, 0 = off
003C     0001        sendisc:rmb      1        ;when 1, indicates DISC packet should be sent
003D     0001        inform: rmb      1        ;when 1, indicates local state has changed,
                                               ;and the distant TIE must be informed
003E     0001        drr:    rmb      1        ;when 1, indicates distant receiver is ready
003F     0001        pouts:  rmb      1        ;when 1, a packet has been sent but not acked
0040     0001        colcnt: rmb      1        ;number of collisions current packet has encountered
0041     0001        tcnt:   rmb      1        ;number of times current packet has been retransmitted
0042     0001        cursent:rmb      1        ;sequence number of last packet successfully sent
0043     0001        cursub: rmb      1        ;sequence number of packet just submitted

0044     0001        rnum:   rmb      1        ;sequence number of last packet successfully received
0045     0001        tnum:   rmb      1        ;sequence number of packet being acked from distant TIE
0046     0001        ackflag:rmb      1        ;acknowledgement flag
0047     0001        nosuccess:rmb    1        ;when 1, indicates that a packet has been submitted
                                               ;but not successfully transmitted yet

0048     0001        curtrbuf:rmb     1        ;pointer to current data transmit buffer
0049     0001        rbufcnt:rmb      1        ;count of free receiver buffers

004A     0001        currcnt:rmb      1        ;current count of chars in receiver buffer being emptied
004B     0001        currbufx:rmb     1        ;bit rep of current receiver buffer being emptied

004C     0001        rbufoutl:rmb     1        ;address of current reciver buffer being
004D     0001        rbufouth:rmb     1        ;emptied by term-write process
004E     0001        rbuf.outp:rmb    1        ;pointer to next char to be unloaded

004F     0001        curtyp:rmb       1        ;current type of packet being transmitted
0050     0001        realtyp:rmb      1        ;if type is ESCAPE, the real type is here

0051     0001        distimer:rmb     1        ;timer for DISC waiting for que to clear
0052     0001        acktimer:rmb     1        ;timer for ack timeout
0053     0001        wdtimer:rmb      1        ;watchdog timer
0054     0001        wdtimcnt:rmb     1        ;used to extend the watchdog timer

0055     0001        rbufpnt:rmb      1        ;contains bit rep of next receiver buffer to check

0056     0001        stopoutp:rmb     1        ;if 1, stop output to term
0057     0001        stopinp:rmb      1        ;if 1, stop input from term
```

```

0058  0001   distaddl:rmb   1      ;address of the distant TIE participating
0059  0001   distaddh:rmb   1      ;in the "connection" sequence


005A  0001   outmessl:rmb   1      ;used to pass the address of the message to
005B  0001   outmessh:rmb   1      ;be written to the user by "outmess:"
005C  0001   messpnt:rmb    1      ;pointer into message string

005D  0002   tabpoint:rmb   2      ;address of table to search

005F  0003   counters:rmb   3

             ;***************************************************************************
0062         termparams:
0062         ;***************************************************************************
0062  0001   rawcook:rmb    1      ;when 1, indicates raw mode
0063  0001   outstatl:rmb   1      ;state of write term from net buffer process
0064  0001   lfcr:   rmb    1      ;when 1, indicates <lf> is mapped to <cr><lf>
0065  0001   echo.off:rmb   1      ;if one, local echoing turned off
0066  0001   HUPflg: rmb    1      ;when 1, no hangup when DTR goes down
0067  0001   transflg:rmb   1      ;when 1, all characters transmitted as is
0068  0001   tab:    rmb    1      ;when 1, tab output as <ht>
0069  0001   edit:   rmb    1      ;when 1, no line editing
006A  0001   map:    rmb    1      ;when 1, no mapping of lf or ht
006B  000E   delays: rmb    14     ;table of chars to delay for
0079  0001   partype:rmb    1      ;type of parity, one of the following

      0000   NONE = 0
      0001   EVEN = 1
      0002   ODD  = 2
      0003   ANY  = 3


007A  0001   savstat: rmb   1      ;temp loc to save previous output state
007B  0001   instatl: rmb   1      ;current state of read term process
007C  0001   savstatl:rmb   1      ;previous state of read term process

007D  0001   outstat2:rmb   1      ;state of write term from echo buffer process
007E  0001   cmdedit:rmb    1      ;char used for deleting chars within the command line
                                   ;this is changed when a "cha dlc <c>" is done


007F  0080   inptab:rmb     128    ;map all functions on input characters



      0290           org    PAGE2 + $090    ;set up ram below transmit buffer 0

0290  0001   CONNstate:rmb  1      ;state of connection sequence, used to resolve
                                   ;call collisons,       if state = BUSY, all other
                                   ;connection requests are ignored

0291  0001   rbu:    rmb    1      ;reverse, underline, blink
0292  0001   parmindx:rmb   1      ;index into parameter list for cursor positioning

```

```
337 0293     0001        ginlen: rmb      1        ;length of GIN response
338 0294     0001        ginstat:rmb      1        ;status of GIN response
339 0295     0001        ginsav: rmb      1        ;temp storage for ginstat
340
341 0296     0001        netstate:rmb     1        ;state of the net transceiver
342                                                ;if state = IDLE, connections may be initiated
343                                                ;or accepted
344
345 0297     0001        cmdpntr:rmb      1        ;index into command line to use
346 0298     0001        tabindx:rmb      1        ;entry position in table
347 0299     0001        entrysiz:rmb     1        ;size of entries in a specific table
348
349 029A     0001        hostconn:rmb     1        ;number of ports on a host
350 029B     0001        conncnt:rmb      1        ;number of ports tried so far
351 029C     0001        curptype:rmb     1        ;current packet type just received
352 029D     0001        goreleas:rmb     1        ;insure buffer gets released to transmitter
353
354 029E     0001        discstat:rmb     1        ;state of the "disconnect" sequence
355
356 029F     0001        sentDISC:rmb     1        ;when 1, indicates a DISC packet already sent
357
358 02A0     0001        tabx:    rmb     1        ;index into tab table
359 02A1     000C        tabs:    rmb     12       ;contains cursor address of tabs on line
360
361 02AD     0001        cmdstate:rmb     1        ;state of input interpretation
362 02AE     0001        cmdbuf.inp:rmb   1        ;input pointer into command buffer
363 02AF     000F        cmdbuf: rmb      15       ;reserve space for TIE commands
364
365 02BE     0001        flstimer:rmb     1        ;flashing LED timer
366
367                                                ;temporary storage locations
368 02BF     0001        temp:    rmb     1        ;used when no waits interfere
369 02C0     0001        templ:   rmb     1        ;(i.e. conflict use)
370 02C1     0001        temp4:   rmb     1        ;used once where they do
371 02C2     0001        tempesc:rmb      1
372
373                      ;ENQackdelay:rmb           1
374                      ;ENQenqcnt:rmb        1
375                      ;ENQXMdelay:rmb 1
376                      ;QBFnorotary:rmb           1       ;when 1, indicates the rotary software is inactive
377                      ;QBFmonaddr:rmb 1          ;when 1, indicates UB is in address-monitor mode
378                      ;QBFqbfpntr:rmb 1          ;pointer for loading QBF
379                      ;QBFdoqbf:        rmb      1       ;when 1, indicates we're sending QBF
380                      ;QBFqbfcnt:       rmb      3       ;running count of QBF packets
381                      ;QBF
382                                                ;next 14 vars make up the report datagram
383                                                ;leave contiguous!!
384 02C3     0003        tottrns: rmb     3        ;total number of successful transmissions during
385                                                ;        connection period
386 02C6     0003        totrecd:rmb      3        ;total packets received
387 02C9     0003        datatrns:rmb     3        ;total data packets transmitted
388 02CC     0003        datarecd:rmb     3        ;total data packets received
389 02CF     0004        dbtrns:rmb       4        ;data bytes transmitted
390 02D3     0004        dbrecd:rmb       4        ;data bytes received
391 02D7     0003        totlost:rmb      3        ;total number of retries over a connection period
392 02DA     0001        maxlost:rmb      1        ;maximum number of retries per given packet
```

```
02DB     0001       lastlost:rmb    1       ;next to max lost if maxlost = 7
02DC     0003       totcol: rmb     3       ;total number of collisions over a connection period
02DF     0001       maxcol: rmb     1       ;maximum number of collisions per given packet
02E0     0001       wdcount:rmb     1       ;number of watchdog timeout packets sent
02E1     0003       contimer:rmb    3       ;connection timer
02E4     0001       tiestat:rmb     1       ;status of TIE equipment
```

```
     E000                      org     $E000


                      ;this process sets up the 2651
                      ;and sets up default conditions for timers

E000 4C 00E0          hdreset:jmp     hdreset         ;force hardware reset


E003 A9 01            reset:  lda     #1              ;come here when all paramaters are to be
E005 85 02                    sta     initall         ;reinitialized

E007 D8               timinit:cld                     ;make sure processor not in decimal mode
E008 A9 7F                    lda     #$7F            ;to disable all interrupts
E00A 8D 0EC0                  sta     ier6522         ;write to interrupt enable reg

E00D A9 FF                    lda     #$FF            ;to set orb to output
E00F 8D 02C0                  sta     ddrb6522        ;put in ddrb

E012 A9 00                    lda     #0              ;to set ora to input
E014 8D 03C0                  sta     ddra6522        ;put in ddra

E017 A9 80                    lda     #$80            ;set timer 1 to oneshot, P87,
                                                      ;timer 2 to oneshot, no SR or latch
E019 8D 0BC0                  sta     acr6522         ;write to aux ctl reg

E01C A9 60                    lda     #$60            ;set up ca and cb for inactive
E01E 8D 0CC0                  sta     pcr6522         ;program ca2 as an input

E021 A9 00                    lda     #$00            ;make sure LED is turned off
E023 8D 00C0                  sta     orb6522

E026 A9 00                    lda     #HARDWD         ;set up indirect pointer to hardware
E028 85 00                    sta     hardwd          ;watchdog location
E02A A9 1C                    lda     #HARDWD > 8
E02C 85 01                    sta     hardwd + 1

                      ;***********************************************************************

                      ;this routine waits until it detects DCD from the user device

E02E A0 00                    ldy     #0              ;make sure hardware watchdog timer is reset
E030 91 00            CDwait: sta     hardwd@[y]      ;while waiting

E032 AD 0110                  lda     a10stat         ;check to see if DCD has come up yet
E035 29 40                    and     #DCD
                      ;QBF     jeq     diagmode        ;if not go into diagnostic mode
                      ;QBF
E037 F0 F7                    beq     CDwait          ;stay in tight loop until it does

                      ;QBF     lda     #0
                      ;QBF     sta     diag            ;else clear diag flag
                      ;QBF
                      ;***********************************************************************

                      ;this routine waits for 1.3 seconds
```

```
                   ;this is to allow all lines to settle, and the terminal to warm up

E039 A2 14                 ldx     #20             ;let the clock tick 20 times
E03B A9 FF         0:      lda     #$FF            ;set the clock to have a tick length of 65 Ms
E03D 8D 08C0               sta     t216522
E040 8D 09C0               sta     t2h6522         ;storing in this location starts the clock

E043 91 00         1:      sta     hardwd@[y]      ;make sure hardware watchdog timer is reset

E045 AD 0DC0               lda     ifr6522         ;check to see if the clock
E048 29 20                 and     #t2f            ;has ticked yet
E04A F0 F7                 beq     1b

E04C CA                    dex                     ;decrement the counter
E04D D0 EC                 bne     0b              ;loop if not done yet

                   ;*************************************************************************

                   ;this routine outputs a canned message
                   ;if the user can read it, he types a good char and the routine ends
                   ;if he cannot he types a "break", which this routine sees as a framing error
                   ;upon detection of the framing error, the routine sets up the 2651 with the
                   ;next speed to try and then starts over

E04F A2 FF                 ldx     #$FF            ;set up initial stack
E051 9A                    txs

E052 AD 0310               lda     ai0cntr         start speed at default in last prom
E055 A9 30                 lda     #$30
E057 8D 0310               sta     ai0cntr         ;reset 2651, turn off DSR

E05A AD B4FD               lda     parity.def      ;get parity from user specific
E05D 8D 0210               sta     ai0mode
E060 AD B3FD               lda     speed.def       ;start at user specified default
E063 8D 0210               sta     ai0mode
E066 A2 08                 ldx     #8              ;in case not right speed, number of tries in table
E068 4C 84E0               jmp     enable          ;enable tranmit and receive and try prompt

E06B A2 08         spdsense:ldx    #8              ;number of speeds in table to try

E06D CA            spdloop:dex                     ;point at next speed to try
E06E 30 FB                 bmi     spdsense        ;loop back to top when done

E070 AD 0310               lda     ai0cntr         ;reset the 2651
E073 A9 30                 lda     #$30            ;turn off DSR until connected, enable transmit
E075 8D 0310               sta     ai0cntr         ;buffer

E078 AD B4FD               lda     parity.def              ;set up character size and parity
E07B 8D 0210               sta     ai0mode
E07E BD 80FC               lda     spdtab[x]       ;get the speed from the table
E081 8D 0210               sta     ai0mode         ;and set the 2651

E084 A9 27         enable: lda     #$27
E086 8D 0310               sta     ai0cntr         ;enable transmit and receive
E089 A0 00                 ldy     #0
E08B AD 0110       1:      lda     ai0stat         ;wait until the transmiter is ready to send
```

```
E08E 29 01                and     #trdy
E090 F0 F9                beq     1b

E092 98                   tya                     ;save message index
E093 A0 00                ldy     #0              ;zero index for indirect
E095 91 00                sta     hardwd@[y]      ;indirect watchdog reset
E097 A8                   tay                     ;restore message index

E098 B9 A0F8              lda     prompt[y]       ;output the canned message
E09B F0 07                beq     2f              ;look for 0 termination

E09D C8                   iny
E09E 8D 0010              sta     a10data
E0A1 4C 8BE0              jmp     1b

E0A4 AD D8FF      2:      lda     SRCaddr
E0A7 20 B3E0              jsr     outaddr         ;output this TIEs address
E0AA AD D9FF              lda     SRCaddr + 1
E0AD 20 B3E0              jsr     outaddr
E0B0 4C DEE0              jmp     2f              ;and go wait for a response

E0B3 8D BF02      outaddr:sta     temp
E0B6 4A                   lsr
E0B7 4A                   lsr
E0B8 4A                   lsr
E0B9 4A                   lsr                     ;output hex byte
E0BA 20 C2E0              jsr     outASCII        ;as two ASCII bytes
E0BD AD BF02              lda     temp
E0C0 29 0F                and     #$0F

E0C2 C9 0A        outASCII:cmp    #10
E0C4 90 03                jcc     1f

E0C6 18                   clc
E0C7 69 07                adc     #7

E0C9 18           1:      clc
E0CA 69 30                adc     #$30

E0CC A8                   tay

E0CD 98           1:      tya
E0CE A0 00                ldy     #0
E0D0 91 00                sta     hardwd@[y]
E0D2 A8                   tay

E0D3 AD 0110              lda     a10stat
E0D6 29 01                and     #trdy
E0D8 F0 F3                jeq     1b

E0DA 8C 0010              sty     a10data
E0DD 60                   rts

E0DE A0 00        2:      ldy     #0              ;make sure the hardware watchdog is reset
E0E0 91 00        3:      sta     hardwd@[y]
```

```
E0E2 AD 0110               lda     ai0stat         ;make sure that DCD is still there
E0E5 29 40                 and     #DCD
E0E7 D0 03                 jeq     timinit         ;if not, reset the tie
E0E9 4C 07E0

E0EC AD 0110               lda     ai0stat         ;wait until there really is a char
E0EF 29 02                 and     #rrdy
E0F1 F0 ED                 beq     3b

E0F3 AD 0110       3:      lda     ai0stat         ;now, check for a framing error
E0F6 29 20                 and     #FRerr          ;if there is not
E0F8 F0 0A                 beq     initALL         ;go on to the rest of the program

E0FA AD 0010               lda     ai0data         ;otherwise, clear the error flag
E0FD 4C 6DE0               jmp     spdloop         ;go do next speed

                   ;QBF;************************************************************************
                   ;QBF
                   ;QBFdiagmode:lda        ai0cntr         ;reset 2651
                   ;QBF    lda     #$A3
                   ;QBF    sta     ai0cntr         ;set into loopback mode
                   ;QBF    lda     #$4E
                   ;QBF    sta     ai0mode
                   ;QBF    lda     #$3E
                   ;QBF    sta     ai0mode
                   ;QBF    lda     #100
                   ;QBF    sta     diag            ;and indicate diagnostic mode
                   ;QBF    sta     initall
                   ;QBF    jmp     initALL
                   ;QBF
                   ;***********************************************************************


                   ;this routine initializes all tables and variables
                   ;this is where you return after a "disconnect" occurs

E100 A9 00         init:   lda     #0              ;make sure the terminal paramters don't
E102 85 02                 sta     initall         ;get reinitialized

E104 A2 FF         initALL:ldx     #$FF
E106 9A                    txs                     ;set up initial stack


E107 A9 00                 lda     #0              ;get ready to zero
E109 A2 00                 ldx     #0              ;out everything above location 2

E10B 9D 0002       1:      sta     PAGE2[x]
E10E E8                    inx                     ;when x wraps back around to 0 -> quit
E10F D0 FA                 bne     1b              ;zero out PAGE2

E111 A6 02                 ldx     initall         ;if terminal paramters are to be initialized
E113 F0 05                 beq     1f

E115 A2 FF                 ldx     #$FF            ;zero all of page 0, by starting at the top
E117 4C 1CE1               jmp     2f              ;and working down

```

```
E11A A2 61       1:      ldx     #termparams - 1 ;else start just below the terminal paramaters

E11C 95 00       2:      sta     PAGE0[x]
E11E CA                  dex                     ;INITIALIZE
E11F E0 02               cpx     #2
                 ;QBF     cpx     #3
E121 D0 F9               bne     2b


E123 A9 02               lda     #2
E125 85 03               sta     curproc         ;set to bottom of table for sched
E127 85 49               sta     rbufcnt         ;initial number of receiver buffers for data

                 ;***********************************************************************

                 ;initialize the stack of each process with the address of the process
                 ;set the value of the sp in the table to 4 away from the address of the stack

E129 A9 E5               lda     #(toterm - 1) > 8       ;initialize stack of write term process
E12B 8D E001             sta     totermsp
E12E A9 BF               lda     #toterm - 1
E130 8D DF01             sta     totermsp - 1
E133 A9 DC               lda     #totermsp - 4
E135 85 07               sta     jobsp + 0

E137 A9 E7               lda     #(fmnet - 1) > 8        ;initialize stack of read net process
E139 8D C001             sta     fmnetsp
E13C A9 BE               lda     #fmnet - 1
E13E 8D BF01             sta     fmnetsp - 1
E141 A9 BC               lda     #fmnetsp - 4
E143 85 08               sta     jobsp + 1

E145 A9 EA               lda     #(tonet - 1) > 8        ;initialize stack of write net process
E147 8D A001             sta     tonetsp
E14A A9 CB               lda     #tonet - 1
E14C 8D 9F01             sta     tonetsp - 1
E14F A9 9C               lda     #tonetsp - 4
E151 85 09               sta     jobsp + 2

                 ;***********************************************************************

                 ;intialize the table which maps the functions to the characters
                 ;on input from the user device

E153 A5 02               lda     initall         ;check to see if these terminal parmaters are
E155 F0 4B               beq     skpterm         ;to be reinitialized first


E157 A2 D7               ldx     #$D7
E159 BD 00FF     0:      lda     DEFTAB[x]       ;load function number value
E15C F0 0B               jeq     1f              ;yes

E15E CA                  dex                     ;decrement counter
E15F BC 00FF             ldy     DEFTAB[x]       ;load function keyvalue
E162 99 7F00             sta     inptab[y]       ;map function to character in table
E165 CA                  dex                     ;decrement default table address counter
```

```
E166 4C 59E1                jmp     0b


E169 A9 08        1:        lda     #BS
E16B 85 7E                  sta     cmdedit         ;delete char function for command mode

E16D A2 00                  ldx     #0
E16F A0 00                  ldy     #0
E171 BD B5FD      1:        lda     parmdef[x]
E174 99 6200                sta     termparams[y]
E177 E8                     inx
E178 C8                     iny
E179 E0 09                  cpx     #PARM.MAX
E17B D0 05                  bne     2f
E17D A0 11                  ldy     #PARM.MAX + 8
E17F 4C 71E1                jmp     1b
E182 E0 0F        2:        cpx     #PARM.MAX + 6
E184 D0 EB                  bne     1b

E186 AD 0310                lda     ai0cntr
E189 AD 0210                lda     ai0mode         ;get parity type
E18C C9 4E                  cmp     #$4E
E18E D0 05                  bne     1f
E190 A2 00                  ldx     #NONE
E192 4C A0E1                jmp     2f
E195 C9 7A        1:        cmp     #$7A
E197 D0 05                  bne     1f
E199 A2 01                  ldx     #EVEN
E19B 4C A0E1                jmp     2f
E19E A2 02        1:        ldx     #ODD
E1A0 86 79        2:        stx     partype
                  ;***********************************************************************

E1A2 A9 1C        skpterm:lda       #$1C            ;high half of addresses to reset receive bufs
E1A4 85 39                  sta     relbufh         ;set up here so only low half has to be
                                                    ;changed by processes

E1A6 A9 00                  lda     #$00            ;make sure LED is
E1A8 8D 00C0                sta     orb6522         ;turned off

E1AB A9 27                  lda     #$27            ;make sure DCD is turned off until
E1AD 8D 0310                sta     ai0cntr         ;connected

E1B0 A9 02                  lda     #PAGE2 > 8      ;get the high half of address of page 2
E1B2 85 60                  sta     counters + 1

E1B4 A9 80                  lda     #128
E1B6 85 14                  sta     echo.free       ;initial number of available slots in echo buf

E1B8 A9 01                  lda     #1
E1BA 85 3B                  sta     fcstat          ;flow control is "on" from device to TIE
E1BC 85 3E                  sta     drr             ;enable transmitter to send data
E1BE 85 55                  sta     rbufpnt         ;point at first rec buffer
E1C0 85 05                  sta     netread         ;make sure net read is always turned on
E1C2 8D BE02                sta     flstimer        ;turn on flash timer

```

```
                      ;ENQ    lda     #2
                      ;ENQ    sta     ackdelay

E1C5 A9 06                    lda     #trstart
E1C7 85 0D                    sta     tin.p           ;set to first data slot in current trans buf

E1C9 A9 00                    lda     #trbuf0         ;get low half of address of 1st trans buffer
E1CB 85 10                    sta     tout.buf        ;to be transmited: store
E1CD 85 0B                    sta     tin.buf         ;to be filled: store

E1CF A9 02                    lda     #trbuf0 > 8     ;get high half of address of 1st trans buffer
E1D1 85 11                    sta     tout.buf + 1    ;to be transmited: store
E1D3 85 0C                    sta     tin.buf + 1     ;to be filled: store

E1D5 A9 AA                    lda     #trbuf0pt       ;load transmit register with pointer to
E1D7 85 48                    sta     curtrbuf        ;transmit buffer 0 (1st to be transmitted)

E1D9 8D 021C                  sta     clrrb0          ;make sure all receive buffers
E1DC 8D 041C                  sta     clrrb1          ;are released
E1DF 8D 061C                  sta     clrrb2

E1E2 AD D8FF                  lda     SRCaddr         ;set up the source address
E1E5 8D 0302                  sta     trbuf0 + srcaddr        ;in each of the transmit buffers
E1E8 8D 0303                  sta     trbuf1 + srcaddr
E1EB 8D 0304                  sta     trbuf2 + srcaddr
E1EE 49 FF                    eor     #$FF            ;put the complement of the source address
E1F0 8D 0018                  sta     TIEaddrh        ;in register for hardware recognition

E1F3 AD D9FF                  lda     SRCaddr + 1             ;finish setting up the buffers
E1F6 8D 0402                  sta     trbuf0 + srcaddr + 1
E1F9 8D 0403                  sta     trbuf1 + srcaddr + 1
E1FC 8D 0404                  sta     trbuf2 + srcaddr + 1
E1FF 49 FF                    eor     #$FF
E201 8D 0118                  sta     TIEaddrl

                      ;QBF    lda     diag            ;if diag mode
                      ;QBF    beq     1f
                      ;QBF
                      ;QBF    lda     #$A3            ;reset the uart
                      ;QBF    sta     ai0cntr         ;back to loop back
                      ;QBF
                      ;QBF    jsr     cnomap          ;put in -map mode
                      ;QBF    jsr     cnoansi         ;put in -ansi mode
                      ;QBF
                      ;QBF1:

E204 A2 7F                    ldx     #127

E206 B5 7F            2:      lda     inptab[x]       ;else find out what the TIE command mode is
E208 29 FE                    and     #$FE            ;strip out break bit
E20A C9 14                    cmp     #fctle          ;is this the attention char?
E20C F0 06                    beq     3f

E20E CA                       dex                     ;if not, keep looking for it
E20F 10 F5                    bpl     2b
E211 4C E7F2                  jmp     setclk
```

```

E214 8E BF02       3:      stx     temp

E217 A0 08                 ldy     #attmess        ;inform the user
E219 A9 FE                 lda     #attmess > 8
E21B 20 9BE7               jsr     outmess

E21E AD BF02               lda     temp            ;of the attention char in effect
E221 20 2CE2               jsr     prntchr
E224 A9 0A                 lda     #LF
E226 20 A7E5               jsr     echoal

E229 4C 87E2               jmp     setclk          ;go start up clock for watchdog and ack timers

                   ;*****************************************************************

E22C C9 20         prntchr:cmp     #$20            ;if char is control-type
E22E B0 0E                 bcs     1f

E230 8D BF02               sta     temp            ;echo it as a two char sequence
E233 A9 5E                 lda     #'©
E235 20 A7E5               jsr     echoal

E238 AD BF02               lda     temp
E23B 18                    clc
E23C 69 40                 adc     #'@
E23E 20 A7E5       1:      jsr     echoal
E241 60                    rts


```

```
                      ;wait allows a process to relinquish the cpu, first saving state
                      ;the 2 index registers are saved along with the program counter at
                      ;the time of the wait, which is pushed on the stack by the "jsr" to this routine

E242 8A               wait:   txa
E243 48                       pha                     ;push index reg x
E244 98                       tya
E245 48                       pha                     ;push index reg y
E246 A4 03                    ldy     curproc
E248 BA                       tsx
E249 96 07                    stx     jobsp[y]        ;store the stack pointer of this process



E24B                  timer:
E24B AD 0DC0                  lda     ifr6522         ;has the clock ticked yet
E24E 29 20                    and     #t2f
E250 F0 3F                    beq     chkterm         ;no, go see if char from term

E252 A5 52                    lda     acktimer        ;yes, check if ack timer is active
E254 F0 02                    beq     1f              ;no, go check watchdog

E256 C6 52                    dec     acktimer        ;decrement clock for ack timeout mechanism

                      ;ENQ1:      lda     XMdelay         ;yes, check if delay timer is active
                      ;ENQ        jeq     1f              ;no, go check watchdog
                      ;ENQ
                      ;ENQ        dec     XMdelay
                      ;ENQ
E258 AD 9002          1:      lda     CONNstate
E25B D0 12                    bne     1f              ;are we connected to anyone?

                      ;QBF    lda     diag
                      ;QBF    beq     2f
                      ;QBF
                      ;QBF    dec     diag
                      ;QBF    jeq     reset
                      ;QBF
                      ;QBF2:

E25D CE BE02                  dec     flstimer        ;if not, flash the LED
E260 D0 0D                    bne     1f

E262 AD 00C0                  lda     orb6522
E265 49 20                    eor     #$20
E267 8D 00C0                  sta     orb6522

E26A A9 08                    lda     #8              ;reset the timer to flash
E26C 8D BE02                  sta     flstimer

E26F A5 51            1:      lda     distimer        ;if DISC timer is active
E271 F0 02                    beq     1f

E273 C6 51                    dec     distimer        ;decrement it

E275 AD 9002          1:      lda     netstate
```

```
E278 C9 18 ´                cmp     #CONNECT
E27A D0 05                  bne     1f              ;if connected

E27C A2 E1                  ldx     #contimer       ;keep time of connection
E27E 20 BDED                jsr     inccount

E281 A5 53          1:      lda     wdtimer         ;is watch dog timer active
E283 F0 02                  beq     setclk          ;no, go reset timer

E285 C6 53                  dec     wdtimer         ;decrement clock for watch dog mechanism

E287 A9 24          setclk: lda     #$24            ;set timer for 62.5ms interval (1/16 sec)
E289 8D 08C0                sta     t216522
E28C A9 F4                  lda     #$F4
E28E 8D 09C0                sta     t2h6522         ;store high half last, also starts timer


E291                chkterm:
E291                ;QBF    lda     doqbf           ;doing qbf?
E291                ;QBF    beq     3f              ;no
E291                ;QBF
E291                ;QBF    lda     ai0stat
E291                ;QBF    and     #rrdy           ;character to service?
E291                ;QBF    beq     4f              ;no
E291                ;QBF
E291                ;QBF    lda     ai0data         ;yes, get character
E291                ;QBF    ldx     partype         ;if parity type none, don't mask char
E291                ;QBF    beq     0f
E291                ;QBF    and     #$7F            ;else strip parity
E291                ;QBF
E291                ;QBF0:  cmp     #CNTLT          ;is it a control-t?
E291                ;QBF    bne     4f              ;if not, ignore it
E291                ;QBF
E291                ;QBF    lda     #0              ;if it is....
E291                ;QBF    sta     doqbf           ;shut off qbf
E291                ;QBF    sta     echo.off        ;and turn on echo
E291                ;QBF    jmp     getchar         ;and go service new command
E291                ;QBF
E291                ;QBF4:  lda     stopinp         ;is input from terminal turned off?
E291                ;QBF    bne     7f              ;if so, then skip over
E291                ;QBF
E291                ;QBF5:  ldx     qbfpntr         ;else, get next qbf character
E291                ;QBF    lda     qbfpckt[x]
E291                ;QBF    bne     6f              ;if not end of qbfpckt
E291                ;QBF
E291                ;QBF    sta     qbfpntr         ;if it is the end, reset pointer
E291                ;QBF    jsr     insrtnum        ;and insert the qbf sequence number
E291                ;QBF    jmp     5b              ;load first char of next qbf
E291                ;QBF
E291                ;QBF6:  inc     qbfpntr         ;increment pointer
E291                ;QBF    ldx     #$0160          ;set up stack for read term process
E291                ;QBF    txs
E291                ;QBF
E291                ;QBF    jsr     qbfentr         ;put next qbf char into trans buffer
E291                ;QBF    jmp     7f
E291                ;QBF
```

```
E291                ;QBF3:
E291 AD 0110                lda     ai0stat         ;see if there is a char to service from term
E294 29 02                  and     #rrdy
E296 D0 2A                  bne     getchar         ;yes, go get character

                    ;QBF7:  lda     diag            ;if diagmode
                    ;QBF    jne     0f              ;don't check for DCD
                    ;QBF
E298 AD 0110                lda     ai0stat         ;else, see if DCD is still there
E29B 29 40                  and     #DCD
E29D D0 29                  bne     0f              ;if it is, just continue

                            ;DCD has gone away---reset only if not connected or
                            ;we are connected and auto hangup is on

E29F AD 9602                lda     netstate        ;else, see if currently connected
E2A2 C9 18                  cmp     #CONNECT
E2A4 F0 03                  jne     hdreset         ;no,reset
E2A6 4C 00E0

                            ;DCD is gone away but we are connected
E2A9 A5 66                  lda     HUPflg          ;yes, auto hangup on?
E2AB D0 1B                  bne     0f              ;no, continue

                            ;DCD is gone, we ARE connected and auto hangup is on
                            ;so disconnect nicely and wait before reset
E2AD AD 9F02                lda     sentDISC        ;else see if we are already attempting
E2B0 D0 1E                  bne     sched           ;a "disconnect" sequence. If so, keep going

E2B2 A9 A0                  lda     #160            ;set disconnect timer for
E2B4 85 51                  sta     distimer        ;approx 10 sec

E2B6 A9 01                  lda     #1
E2B8 85 3C                  sta     sendisc         ;set flag to send the DISC packet
E2BA 85 06                  sta     netwrite        ;and to activate the net write process
E2BC 8D 9F02                sta     sentDISC        ;indicate that this is being done
E2BF 4C D0E2                jmp     sched           ;go schedule transmitter

E2C2 A2 60          getchar:ldx     #$0160
E2C4 9A                     txs                     ;set up a stack for read term process
E2C5 20 F2E2                jsr     fmterm          ;go get char

E2C8 AD 9D02        0:      lda     goreleas        ;is there a packet waiting to be transmitted
E2CB F0 03                  beq     sched           ;no, go schedule the next process

E2CD 20 F3E3                jsr     release         ;yes, go see if it can be sent yet

                    ;*****************************************************************************



                    ;sched schedules the next process in the table that is runnable
                    ;at present this is done strictly round-robin
                    ;it indexes through the table of things to do backwards

E2D0 A4 03          sched:  ldy     curproc
E2D2 88             2:      dey                     ;get next process number
```

```
E2D3 10 02                 bpl     1f              ;if neg, reset to top of table
E2D5 A0 02                 ldy     #PROCMAX        ;get maximum process number
E2D7 B9 0400       1:      lda     jobstat[y]      ;get status of process to be scheduled
E2DA F0 F6                 beq     2b              ;if zero, try next process in table
E2DC 84 03                 sty     curproc         ;update current process id


                   ;************************************************************************

                   ;dispat restores the 2 index registers, and sets the program counter
                   ;to the address in the process where the wait occured

E2DE B6 07         dispat: ldx     jobsp[y]        ;get stack pointer for this process

E2E0 A0 00                 ldy     #0              ;make sure hardware watchdog
E2E2 91 00                 sta     hardwd@[y]      ;is reset

E2E4 9A                    txs                     ;set stack to the stack pointer associated
                                                   ;with this process
E2E5 68                    pla
E2E6 A8                    tay                     ;restore index reg y
E2E7 68                    pla
E2E8 AA                    tax                     ;restore index reg x
E2E9 60                    rts                     ;return to where wait occured

                   ;************************************************************************


                           nbin
E2EA               nxtindx:byte    1,2,3,4,5,6,7,0
                   ;QBFqbfmess:byte         "000000  the quick brown fox jumped over the lazy dogs back",LF,0
```

```
                        ;routines to interpret input from terminal


     E2F2               fmterm:
     E2F2 AD 0110               lda     ai0stat         ;check for framing error
     E2F5 29 20                 and     #FRerr
     E2F7 F0 16                 jeq     1f

     E2F9 A2 37                 ldx     #$37            ;clear error bits
     E2FB 8E 0310               stx     ai0cntr

     E2FE AD 0010               lda     ai0data         ;get character
     E301 C9 00                 cmp     #0              ;is it a null(i.e, was the break key hit)?
     E303 D0 09                 jne     2f

     E305 A5 7B                 lda     instatl         ;get current input state
     E307 C9 03                 cmp     #3              ;cmd edit mode
     E309 F0 03                 jeq     2f
     E30B 4C 7EE4               jmp     xctie

     E30E 60            2:      rts
     E30F AD 0010       1:      lda     ai0data         ;get character
                        ;LAN    jpl     1f              ;if high orer bit is not set, just continue
                        ;LAN
                        ;LAN    ldx     transflg        ;else see if maybe in transparent mode
                        ;LAN    jeq     fmtermrl


     E312 29 7F         1:      and     #$7F
                        ;TKX    ldx     ginstat         ;if character could be in response to GIN
                        ;TKX    cpx     #3
                        ;TKX    jcs     respgin         ;then go check

                        ;QBFqbfentr:ldx monaddr         ;is address monitor active?
                        ;QBF    beq     normal          ;no, skip
                        ;QBF
                        ;QBF    tax                     ;if yes,
                        ;QBF    lda     inptab[x]       ;only look for control keystrokes
                        ;QBF    and     #$7E
                        ;QBF
                        ;QBF    cmp     #fctie          ;attention char
                        ;QBF    jeq     hdreset
                        ;QBF
                        ;QBF    cmp     #fcstr          ;start output to term
                        ;QBF    jeq     xcstr
                        ;QBF
                        ;QBF    cmp     #fcstp          ;stop output to term
                        ;QBF    jeq     xcstp
                        ;QBF    rts                     ;else, ignore the character
                        ;QBF

     E314 A6 7B         normal: ldx     instatl         ;get input state
     E316 D0 03                 jeq     state0          ;go do state0 (cooked)
     E318 4C ACE3

     E31B E0 01                 cpx     #1
```

```
E31D F0 7C                 jeq     state1           ;go do state1 (raw)

E31F E0 02                 cpx     #2
E321 F0 6D                 jeq     state2           ;go do state2 (escape type)

                   ;************************************************************************

                                                    ;must be state3, COMMAND INTERPRETER
E323               state3:
E323 AE AD02               ldx     cmdstate         ;is state of command input escape
E326 F0 08                 jeq     1f               ;or normal interpretation

E328 A2 00                 ldx     #0
E32A 8E AD02               stx     cmdstate         ;reset command input state

E32D 4C 77E3               jmp     2f               ;just put character in buffer

E330               1:
E330 C9 1B                 cmp     #ESC             ;is it an escape?
E332 D0 06                 jne     1f               ;no

E334 A2 01                 ldx     #1               ;set command input state for escape
E336 8E AD02               stx     cmdstate
E339 60                    rts

E33A               1:
E33A C5 7E                 cmp     cmdedit          ;is it the delete char command
E33C D0 13                 jne     1f               ;no

E33E AE AE02               ldx     cmdbuf.inp       ;are we at the beginning of the buffer
E341 F0 4C                 jeq     3f               ;yes, ignore

E343 A9 FF                 lda     #bsstr > 8       ;point to BS,SP,BS string, MSB = 1
E345 20 A7E5               jsr     echoal           ;otherwise erase last character from screen
E348 A9 59                 lda     #bsstr
E34A 20 A7E5               jsr     echoal

E34D CE AE02               dec     cmdbuf.inp       ;dec input pointer of command buffer
E350 60                    rts                      ;return

                   ;************************************************************************

E351 C9 0D         1:      cmp     #CR              ;is it a <CR>
E353 D0 02                 jne     1f               ;no

E355 A9 0A                 lda     #LF              ;yes, map into a <LF>
E357 20 A7E5       1:      jsr     echoal           ;echo the char
E35A C9 0A                 cmp     #LF              ;was it a <LF>
E35C D0 19                 jne     2f               ;no, go store char away

E35E A5 7C                 lda     savstat1         ;get saved input state
E360 85 7B                 sta     instat1          ;restore input state1(from term)
E362 AE AE02               ldx     cmdbuf.inp
E365 F0 08                 jeq     5f               ;if no chars in buf, just return

E367 A9 00                 lda     #0               ;end command string with a zero
```

```
E369 9D AF02                sta    cmdbuf[x]

E36C 20 57EE                jsr    cmdintrp        ;go interpret command string

E36F A9 00         5:       lda    #0
E371 8D AE02                sta    cmdbuf.inp      ;put input pointer back to beginning of buffer
E374 85 56                  sta    stopoutp        ;turn on the output from the network
E376 60                     rts

                   ;**********************************************************************

E377 AE AE02       2:       ldx    cmdbuf.inp      ;get current input pointer
E37A E0 0E                  cpx    #14             ;are we at end of buffer
E37C F0 11                  jeq    3f              ;yes, ignore char and return

E37E C9 41                  cmp    #'A             ;if the character is a capital letter,
E380 90 07                  jcc    0f
E382 C9 5B                  cmp    #'[
E384 B0 03                  jcs    0f

E386 18                     clc
E387 69 20                  adc    #$20            ;make it the corresponding small letter

E389 9D AF02       0:       sta    cmdbuf[x]       ;store char away
E38C EE AE02                inc    cmdbuf.inp      ;bump up input pointer
E38F 60            3:       rts                    ;return to schedule next runnable process

                   ;**********************************************************************


                                                   ;type is ESCAPE
E390               state2:
E390 A6 67                  ldx    transflg
E392 F0 41                  jeq    fmtermr0
E394 A6 7C                  ldx    savstat1        ;get stored input state
E396 86 7B                  stx    instat1         ;restore input state of term
E398 4C D5E3                jmp    fmtermr0        ;pass through char unmolested

                   ;**********************************************************************


                                                   ;RAW MODE       -LINE MODE
E39B               state1:
E39B A6 67                  ldx    transflg
E39D F0 36                  jeq    fmtermr0
E39F AA                     tax                    ;save the character
E3A0 B5 7F                  lda    inptab[x]       ;see what function it asks for
E3A2 29 7E                  and    #$7E            ;get rid of break indication
E3A4 C9 0A                  cmp    #10             ;don't do any editting features
E3A6 10 23                  jpl    3f              ;go interpret other commands
E3A8 8A                     txa
E3A9 4C D5E3                jmp    fmtermr0        ;echo all these characters

                   ;**********************************************************************

                                                   ; LINE MODE
                                                   ;COOKED MODE
E3AC               state0:
```

```
E3AC AA                     tax                     ;acc -> x-reg

E3AD B5 7F                  lda     inptab[x]       ;see if char is break class char
E3AF 29 01                  and     #BREAK
E3B1 85 0F                  sta     tin.brk         ;set break flag

E3B3 A4 67                  ldy     transflg        ;transparent mode?
E3B5 D0 04                  bne     4f              ;no

E3B7 8A                     txa                     ;yes, don't interpret
E3B8 4C D5E3                jmp     fmtermr0
E3BB                4:
E3BB B5 7F                  lda     inptab[x]       ;get index into table of addresses
E3BD 29 7E                  and     #$7E            ;get rid of bit which indicates break class
E3BF A4 69                  ldy     edit            ;is line editting feature enabled?
E3C1 F0 08                  jeq     3f              ;yes

E3C3 C9 0A                  cmp     #10             ;no, don't do rpl, dll, dlw, and dlc commands
E3C5 10 04                  jpl     3f              ;go interpret other commands
E3C7 8A                     txa
E3C8 4C D5E3                jmp     fmtermr0        ;echo all these characters

E3CB A8             3:      tay

E3CC B9 57E4                lda     cntrfnc+1[y]    ;get high half of address of function
E3CF 48                     pha                     ;push on stack for rts
E3D0 B9 56E4                lda     cntrfnc[y]      ;get low half
E3D3 48                     pha                     ;push on stack for rts
E3D4 60                     rts                     ;transfer control to function routine

                    ;************************************************************************

E3D5 20 9FE5        fmtermr0:jsr    echo            ;echo character
E3D8                fmtermr1:
E3D8 A4 0E                  ldy     tin.cnt         ;allow a few more chars
E3DA C0 55                  cpy     #LINSIZ+5       ;before ignoring them after full buffer
E3DC F0 15                  jeq     release         ;in case HPterm can't stop on the char

E3DE A4 0D                  ldy     tin.p           ;get current in pointer
E3E0 91 0B                  sta     tin.buf@[y]     ;store char in current trans buf
E3E2 E6 0D                  inc     tin.p           ;bump current in pointer
E3E4 E6 0E                  inc     tin.cnt         ;bump current count of chars
E3E6 A5 0E                  lda     tin.cnt
E3E8 C9 50                  cmp     #LINSIZ
E3EA F0 07                  jeq     release

E3EC A5 0F                  lda     tin.brk         ;was char a break class char
E3EE 05 62                  ora     rawcook         ;or, are we in raw mode
E3F0 D0 01                  jne     release         ;yes, go release buffer to net-write rout

E3F2 60                     rts                     ;return

                    ;************************************************************************

E3F3 AD 9602        release:lda     netstate        ;get state of network side
E3F6 C9 18                  cmp     #CONNECT        ;are we connected
```

```
E3F8 D0 39              jne    4f              ;no, reinitialize buf pointers and return

E3FA A5 0A              lda    tran.used       ;how many used trans buffers are there
E3FC F0 18              jeq    3f              ;if there are 0, go ahead and release this one

E3FE A9 01              lda    #1              ;else set flag
E400 8D 9D02            sta    goreleas        ;that indicates buf to be released

E403 A5 62              lda    rawcook         ;if line mode,
E405 F0 04              jeq    1f              ;stop input from term

E407 A5 0E              lda    tin.cnt         ;if -line mode, is buffer full yet?
E409 10 0A              jpl    2f              ;no, go return

E40B A9 01      1:      lda    #1              ;set flag to
E40D 85 57              sta    stopinp         ;turn off the input routine

E40F A9 02              lda    #2              ;drop flow control line
E411 85 3A              sta    fctask
E413 85 04              sta    outterm

                ;QBF    lda    diag
                ;QBF    beq    2f
                ;QBF    sta    stopoutp
                ;QBF
E415 60         2:      rts

E416 E6 0A      3:      inc    tran.used       ;increment count of used trans buffers
E418 A5 0E              lda    tin.cnt         ;get count of chars in this trans buf

E41A A2 CF              ldx    #dbtrns         ;add it to the total data bytes trans
E41C 20 BFED            jsr    addcount
E41F A5 0E              lda    tin.cnt         ;restore tin.cnt to acc

E421 18                 clc
E422 69 05              adc    #HEADSIZ        ;allow for 6 byte header
E424 A0 00              ldy    #hardcnt        ;get index of count field
E426 91 0B              sta    tin.buf@(y)     ;store count in trans buf
E428 A5 0C              lda    tin.buf + 1     ;get high half of address of cur trans buf
E42A 49 01              eor    #1              ;change address to point to next trans buf
E42C 85 0C              sta    tin.buf + 1     ;store it away (uses buffs 2 and 3 alternately)

E42E A9 01              lda    #DATA           ;que up data packet for net transmitter
E430 20 46E4            jsr    quepack

E433 A9 00      4:      lda    #0              ;get ready to zero out
E435 85 0E              sta    tin.cnt         ;zero real char count
E437 85 0F              sta    tin.brk         ;the break now test
E439 8D A002            sta    tabx            ;the tab field index
E43C 85 57              sta    stopinp         ;turn on the char input routine
E43E 8D 9D02            sta    goreleas        ;turn off flag to force release

E441 A9 [illegible]     lda    #trstart        ;get the loc of first free slot in new trans buf
E443 85 [illegible]     sta    tin.p           ;store it away in current input pointer

                ;QBF    lda    diag
```

```
                  ;QBF    jne     xcstr
                  ;QBF
E445 60                   rts

                  ;***********************************************************************

                  ;this routine sticks types of packets on a que for the net transmitter
                  ;the packet type is passed in the accumalator

E446 A2 01        quepack:ldx     #1              ;make sure
E448 86 06                stx     netwrite        ;the net transmitter is turned on

E44A A6 28                ldx     queinp          ;get input pointer into que
E44C 95 20                sta     que[x]          ;store packet type on que
E44E E6 1F                inc     quecnt          ;bump up count of things on que
E450 BC EAE2              ldy     nxtindx[x]      ;easy way to wrap pointer around when necessary
E453 84 28                sty     queinp          ;store it away
E455 60                   rts                     ;return

                  ;***********************************************************************

                                                  ;table of addresses of control functions
E456              cntrfnc:
E456                      addr    xcnop-1,xcdlw-1,xcdlc-1,xcdll-1         ; 0  2  4  6
E45E                      addr    xcrpl-1,xctnl-1,xcstp-1,xcstr-1         ; 8  A  C  E
E466                      addr    xcint-1,xcesc-1,xctie-1                 ;10 12 14

                  ;***********************************************************************

E46C A9 00        xcstr:  lda     #0              ;continue output to term from network
E46E 85 56                sta     stopoutp
E470 A9 01                lda     #1
E472 85 04                sta     outterm
E474 60                   rts

                  ;***********************************************************************

E475 A5 7B        xcesc:  lda     instatl
E477 85 7C                sta     savstatl        ;save current state so it can be restored
E479 A9 02                lda     #2              ;set input state to state2 so that
E47B 85 7B                sta     instatl         ;next char will be passed through unmolested
E47D 60                   rts                     ;return and wait for next char

                  ;***********************************************************************

E47E A0 3B        xctie:  ldy     #mess3          ;indicate to user that he is now
E480 A9 FE                lda     #mess3 > 8
E482 20 9BE7              jsr     outmess         ;in the TIE command mode

E485 A5 7B                lda     instatl
E487 85 7C                sta     savstatl        ;save current input state
E489 A9 03                lda     #3              ;set to state3, TIE command to be accepted
E48B 85 7B                sta     instatl

E48D A9 00                lda     #0
E48F 8D AD02              sta     cmdstate
```

```

                   ;QBF    lda  ·  diag            ;if diagnostic mode
                   ;QBF    jne     lf              ;don't turn of the command input
                   ;QBF
                   ;**********************************************************************

E492 A9 01         xcstp:  lda     #1              ;suspend output to term from network
E494 85 56                 sta     stopoutp
                   ;QBF1:
E496 60                    rts

                   ;**********************************************************************

E497 8A            xcnop:  txa                     ;transfer char to acc
E498 4C D5E3               jmp     fmtermr0        ;go store in trans and echo bufs, then return

                   ;**********************************************************************

E49B A9 0A         xctnl:  lda     #LF             ;map cr or lf into a lf
E49D 4C D5E3               jmp     fmtermr0        ;go store in trans and echo bufs, then return

                   ;**********************************************************************
                   ;interrupt

E4A0 20 E3E9       xcint:  jsr     clrtrbfs        ;clear all the transmit buffers

E4A3 8A                    txa                     ;transfer char to acc
E4A4 20 9FE5               jsr     echo            ;put in echo buf
E4A7 A9 0A                 lda     #LF             ;follow it with a lf
E4A9 20 9FE5               jsr     echo

E4AC A9 80                 lda     #INTR1          ;send an interrupt packet out on the NET
E4AE 4C 46E4               jmp     quepack         ;"quepack" will do the return

                   ;**********************************************************************

E4B1 A5 0E         xcrpl:  lda     tin.cnt         ;<repeat line function>
E4B3 F0 11                 jeq     return          ;see if there are any chars currently in buf
                                                   ;if not just go return
E4B5 A9 0A                 lda     #LF             ;else put out a <LF>
E4B7 20 9FE5               jsr     echo

E4BA A0 06                 ldy     #trstart        ;followed by the characters in the buffer
E4BC B1 0B         1:      lda     tin.buf@[y]
E4BE 20 9FE5               jsr     echo
E4C1 C8                    iny
E4C2 C4 0D                 cpy     tin.p
E4C4 D0 F6                 bne     1b

E4C6 60            return: rts

                   ;**********************************************************************

                                                   ; delete character
E4C7 20 7BE5       xcdlc:  jsr     backup          ;find out how many cursor pos to back up
E4CA 4C 3CE5               jmp     decide          ;decide on best way, then do it
```

```

                  ;************************************************************************

                  ;delete word

E4CD A5 0E        xcdlw:  lda     tin.cnt         ;see if any chars in trans buf
E4CF F0 F5                jeq     return

E4D1 A9 00                lda     #0
E4D3 8D C002              sta     temp1           ;zero temp loc
E4D6 A4 0D                ldy     tin.p           ;get current input pointer

E4D8 88           1:      dey                     ;point pointer at real char
E4D9 B1 0B                lda     tin.buf@[y]     ;get char
E4DB C9 20                cmp     #SP             ;is it a space
E4DD F0 04                jeq     4f              ;yes, continue
E4DF C9 09                cmp     #HT             ;no, is it a tab
E4E1 D0 1D                jne     2f              ;no, go do next part
E4E3 8C BF02      4:      sty     temp            ;save current pointer, y gets clobbered
E4E6 20 7BE5              jsr     backup          ;find out how many cursor pos to back up
E4E9 F0 2D                jeq     3f              ;if zero, finished
E4EB 18                   clc                     ;else, clear the carry
E4EC 6D C002              adc     temp1           ;add this amount to total
E4EF 8D C002              sta     temp1           ;update total
E4F2 AC BF02              ldy     temp            ;restore input pointer
E4F5 4C D8E4              jmp     1b              ;go check for more spaces or tabs

E4F8 C9 20        0:      cmp     #SP             ;is it a space
E4FA F0 1C                jeq     3f              ;yes, finished
E4FC C9 09                cmp     #HT             ;is it a tab
E4FE F0 18                jeq     3f              ;yes, finished
E500 8C BF02      2:      sty     temp            ;save current input pointer
E503 20 7BE5              jsr     backup
E506 F0 10                beq     3f              ;if zero, finished
E508 18                   clc                     ;else clear carry
E509 6D C002              adc     temp1           ;add this amount to total
E50C 8D C002              sta     temp1           ;update total
E50F AC BF02              ldy     temp            ;restore input pointer
E512 88                   dey                     ;point to next char
E513 B1 0B                lda     tin.buf@[y]     ;get next char
E515 4C F8E4              jmp     0b              ;continue while not a space or tab

E518 AD C002      3:      lda     temp1           ;get final total of cursor pos to back up
E51B 4C 3CE5              jmp     decide          ;go decide best way, then do

                  ;************************************************************************

E51E A9 00        xcdll:  lda     #0              ;<delete line function>
E520 8D C002              sta     temp1
                                                  ;find out how many cursor positions to back up
E523 A5 0E        1:      lda     tin.cnt         ;if no more chars in buf, continue
E525 F0 0D                jeq     2f

E527 20 7BE5              jsr     backup          ;else find out how many cursor positions
                                                  ;current char takes up
E52A 18                   clc
```

```
E52B 6D C002               adc     templ           ;keep a running additive count
E52E 8D C002               sta     templ

E531 4C 23E5               jmp     1b

E534 A9 00         2:      lda     #0
E536 8D A002               sta     tabx            ;clear index into tab table
E539 AD C002               lda     templ

                   ;*********************************************************************

                                                   ;always move cursor with relative
                                                   ;positioning, first CUB the right number
                                                   ;of positions, then ECH that many positions
                                                   ;to get rid of characters.
E53C F0 16         decide: jeq     4f              ;if 0, no cursor postions to back up
                                                   ;just go return

E53E 8D BF02               sta     temp            ;save number of positions
E541 20 55E5               jsr     escpl           ;go set up escape sequence
E544 A9 44                 lda     #'D             ;CUB - code for relative backward cursor
                                                   ;       positioning
E546 20 9FE5       2:      jsr     echo            ;go store in echo buf


E549 AD BF02               lda     temp            ;get back number of positions
E54C 20 55E5               jsr     escpl           ;start escape sequence
E54F A9 58                 lda     #'X             ;do ECH that many places
E551 20 9FE5               jsr     echo

E554 60            4:      rts

                   ;*********************************************************************

                   ;this routine generates an escape sequence for the term write routine to
                   ;utilize, it must first convert the paramter to ASCII decimal

E555 A2 00         escpl:  ldx     #0              ;use x reg as counter, zero first
E557 C9 0A         0:      cmp     #10             ;is acc < 10
E559 90 05                 jcc     1f              ;yes, finished
E55B E9 0A                 sbc     #10             ;no, subtract 10
E55D E8                    inx                     ;inc tens counter
E55E D0 F7                 jne     0b              ;continue
E560 18            1:      clc                     ;clear carry
E561 69 30                 adc     #'0             ;make ASCII
E563 48                    pha                     ;store on stack
E564 8A                    txa                     ;get tens counter
E565 18                    clc
E566 69 30                 adc     #'0             ;make ASCII
E568 48                    pha                     ;store on stack

E569 A9 FF                 lda     #escstr > 8
E56B 20 9FE5               jsr     echo            ;go put in echo buf
E56E A9 54                 lda     #escstr
E570 20 9FE5               jsr     echo
```

```
E573 68                    pla                     ;pull first ASCII number off of stack
E574 20 9FE5               jsr     echo            ;go put in echo buf
E577 68                    pla                     ;pull second ASCII numb off of stack
E578 4C 9FE5               jmp     echo            ;go put in echo buf

                  ;**********************************************************************

                  ;this routine returns the number of cursor positions to back up for each
                  ;character, it also updates the pointer and counter associated with the
                  ;current transmit buffer:
                                   ;returns 0 - if no more chars in trans buf, or non-printing char
                                   ;returns 1 - if normal char
                                   ;returns 2 - if printing control char
                                   ;returns n - where n is number of postions to back up over tab

E57B A5 0E        backup:  lda     tin.cnt         ;get current input char count
E57D F0 1F                 jeq     3f              ;if zero, return zero

E57F C6 0D                 dec     tin.p           ;point pointer at real char
E581 C6 0E                 dec     tin.cnt         ;decrement count
E583 A4 0D                 ldy     tin.p           ;put pointer in index reg y
E585 B1 0B                 lda     tin.buf@[y]     ;get char

E587 C9 20                 cmp     #SP             ;is it a non-control char
E589 B0 07                 jcs     1f              ;yes,return 1

E58B C9 09                 cmp     #HT             ;is it a horizontal tab
E58D F0 06                 jeq     2f              ;yes, go do tab lookup

E58F A9 00                 lda     #0              ;return 0
E591 60                    rts

E592 A9 01        1:       lda     #1              ;return 1
E594 60                    rts


E595 CE A002      2:       dec     tabx            ;point tab table index at current tab
E598 AE A002               ldx     tabx            ;load index
E59B BD A102               lda     tabs[x]         ;get number of cursor positions it generated
E59E 60           3:       rts                     ;return this number

                  ;**********************************************************************

                  ;echo: put a char in the terminal echo buffer unless the echo is turned off
                  ;        or there is no room in the transmit buffer for the character
                  ;echoal: always put a char in the terminal echo buffer
                  ;        the character is passed in the accumulator, in both cases

E59F              echo:
E59F              ;QBF      ldx     diag            ;if diagnostic mode
E59F              ;QBF      bne     2f              ;don't echo
E59F              ;QBF
E59F A6 65                 ldx     echo.off        ;check to see if echo
E5A1 D0 1C                 jne     2f              ;has been turned off
E5A3 A6 57                 ldx     stopinp         ;or if input from the terminal has been
E5A5 D0 18                 jne     2f              ;halted. If so, lose char
```

```

E5A7                    echoal:
E5A7                    ;QBF    ldx     diag            ;if diagnostic mode
E5A7                    ;QBF    bne     2f              ;don't echo
E5A7                    ;QBF
E5A7 A6 14                      ldx     echo.free       ;get number of free slots
E5A9 F0 14                      jeq     2f              ;if 0, lose char

E5AB A6 12                      ldx     echo.inp        ;get current echo buf inp pointer
E5AD 9D 3004                    sta     echo.start[x]   ;store the char away
E5B0 C6 14                      dec     echo.free       ;decrement the number of free slots
E5B2 E6 15                      inc     echo.used       ;bump up echo buf char count
E5B4 E8                         inx                     ;bump up the input pointer
E5B5 10 02                      jpl     1f              ;if it is not at the end of the buf, go store

E5B7 A2 00                      ldx     #0              ;else, point to beginning of echo buf
E5B9 86 12              1:      stx     echo.inp        ;store away
E5BB A2 01                      ldx     #1
E5BD 86 04                      stx     outterm         ;make sure term write routine gets scheduled
E5BF 60                 2:      rts                     ;return



                        ; 790910 RJC
                        ;routines to write to terminal, must be combined with XXX-write.s65
                        ;zroutines which contain terminal-specific cursor positioning
                        ; routines, even for BS, etc.


                        ;this routine unloads both the echo buffer and the buffer filled from network
                        ;the echo buffer is always unloaded first, then the network buffer
                        ;a wait is done between each char unloaded


E5C0 A5 3A              toterm: lda     fctask
E5C2 F0 03                      jeq     0f              ;if flow control task, go do it

E5C4 20 A3E7                    jsr     fcterm

E5C7 A5 15              0:      lda     echo.used       ;are there any chars in the echo buf
E5C9 F0 62                      beq     chknet          ;no, go check for data from net

E5CB A6 13              1:      ldx     echo.outp       ;get echo output pointer
E5CD BD 3004                    lda     echo.start[x]   ;get char out of echo buf
E5D0 30 23                      bmi     string          ;see if the high order bit is turned on
                                                        ;if not, then it is just a regular char
                                                        ;from the keyboard

E5D2 E8                         inx                     ;bump up output pointer
E5D3 10 02                      bpl     1f              ;>- 128 ?

E5D5 A2 00                      ldx     #0              ;yes, wrap it around to start of echo buf
E5D7 86 13              1:      stx     echo.outp       ;store away

E5D9 AA                         tax
E5DA C9 1B                      cpx     #ESC            ;if char is ESC
```

```
E5DC D0 06                 bne     0f

E5DE 20 0AE7               jsr     putout          ;just echo it
E5E1 4C EBE5               jmp     1f

E5E4 A5 7D         0:      lda     outstat2        ;partial interpretation of chars from
E5E6 20 8FE6               jsr     interp          ;keyboard
E5E9 85 7D                 sta     outstat2

E5EB E6 14         1:      inc     echo.free       ;free a slot in echo buf
E5ED C6 15                 dec     echo.used       ;decrement char count

E5EF 20 42E2               jsr     wait            ;relinquish control for awhile
E5F2 4C C0E5               jmp     toterm          ;when restored start over

                   ;***********************************************************************

                                                   ;high order bit being set, indicates that
                                                   ;this byte and the next are really the
                                                   ;address of a canned message to be output
E5F5 85 5B         string: sta     outmessh        ;the high half of the address comes first
E5F7 E8                    inx
E5F8 10 02                 bpl     3f
                                                   ;do the normal wrap around for a circular
E5FA A2 00                 ldx     #0              ;buffer
E5FC BD 3004       3:      lda     echo.start[x]   ;get the next char,
E5FF 85 5A                 ata     outmessl        ;and store it as the low half of the address
E601 E8                    inx
E602 10 02                 bpl     3f
                                                   ;have to check for wrap around here also
E604 A2 00                 ldx     #0
E606 86 13         3:      stx     echo.outp

E608 A2 00         putmess:ldx     #0              ;pointer into message being output
E60A 86 5C                 stx     messpnt

E60C A4 5C         waitmess:ldy    messpnt         ;get the next char to output
E60E B1 5A                 lda     outmessl@[y]
E610 F0 10                 beq     4f              ;if it is a 0, then end of message

E612 AA                    tax
E613 A5 7D                 lda     outstat2        ;else output the char
E615 20 8FE6               jsr     interp          ;full interpretation is done on it for
E618 85 7D                 sta     outstat2        ;standard cursor positioning
E61A E6 5C                 inc     messpnt

E61C 20 42E2               jsr     wait            ;relinquish control for a while
E61F 4C 0CE6               jmp     waitmess        ;when you get it back, go check for next
                                                   ;char in message
E622 E6 14         4:      inc     echo.free       ;come here when message is done,
E624 E6 14                 inc     echo.free       ;and clean up counters associated
E626 C6 15                 dec     echo.used       ;with the echo buf
E628 C6 15                 dec     echo.used       ;free up 2 slots

E62A 4C C0E5               jmp     toterm          ;go check for more chars in echo buf

```

```
                       ;*****************************************************************

E62D A5 56             chknet: lda     stopoutp        ;output from net turned off ?
E62F D0 54                     bne     nooutput        ;yes, go see if there are any local chars

E631 A5 4A                     lda     current         ;any more chars in cur net buf
E633 D0 33                     bne     1f              ;yes, go get next char

E635 A5 2C                     lda     reccnt          ;no, are there any more net bufs
E637 F0 4C                     beq     nooutput        ;no, go check echo buf

E639 A6 2B                     ldx     recoutp         ;get pointer to next net buf index
E63B B4 2D                     ldy     recque[x]       ;get next net buf index
E63D 84 4B                     sty     currbufx        ;save it for later use
E63F B9 0CEA                   lda     rbufadh[y]      ;get high half of net buf address
E642 85 4D                     sta     rbufouth        ;store in pointer on page 0
E644 BC EAE2                   ldy     nxtindx[x]      ;use cur index to find next index
E647 84 2B                     sty     recoutp         ;store away
E649 C6 2C                     dec     reccnt          ;dec count of cur full net bufs

E64B A0 00                     ldy     #hardcnt        ;load y with offset into count field
E64D B1 4C                     lda     rbufoutl@[y]    ;get count of chars in this buf
E64F 38                        sec
E650 E9 05                     sbc     #HEADSIZ        ;subtract headsize
E652 D0 09                     bne     2f              ;got some real chars

E654 20 11EA                   jsr     relrbuf         ;go release back to net

E657 20 42E2                   jsr     wait            ;release control
E65A 4C C0E5                   jmp     toterm          ;when you get it back, go check for another buf


E65D 85 4A             2:      sta     current         ;store it in cur count of chars for this buf

E65F A2 D3                     ldx     #dbrecd         ;add count to total data bytes received
E661 20 BFED                   jsr     addcount

E664 A0 06                     ldy     #HEADSIZ + 1    ;get the header size + 1 of this packet
E666 84 4E                     sty     rbuf.outp       ;initialize pointer into data field

E668 A4 4E             1:      ldy     rbuf.outp       ;get pointer to next char
E66A B1 4C                     lda     rbufoutl@[y]    ;get next char to output
E66C E6 4E                     inc     rbuf.outp       ;bump up pointer
E66E C6 4A                     dec     current         ;dec cur char count
E670 D0 05                     bne     3f              ;if non-zero skip next sect

E672 48                        pha                     ;save char

E673 20 11EA                   jsr     relrbuf         ;release the buf
E676 68                        pla                     ;restore the char

E677 AA                3:      tax                     ;put char in x-reg
                       ;TKX    jsr     chkgin          ;process GIN request if appropriate
                       ;TKX    sta     ginstat
E678 A5 E3                     lda     outstatl        ;get state of net output rout
E67A 20 8FF6                   jsr     interp          ;go do char interpretation and output
```

```
E67D 85 63                     sta     outstatl          ;save new state
                     ;ENQ      inc     enqcnt            ;increment count of characters since
                     ;ENQ      lda     enqcnt            ;last enq was sent
                     ;ENQ      cmp     #80               ;send enq after 80 characters
                     ;ENQ      jne     1f
                     ;ENQ
                     ;ENQ      ldx     #ENK              ;if so, send enq and stop output
                     ;ENQ      jsr     putout
                     ;ENQ      lda     #1
                     ;ENQ      sta     stopoutp
                     ;ENQ      lda     #0
                     ;ENQ      sta     enqcnt            ;set char count for enq back to zero
                     ;ENQ
                     ;ENQ      ldx     ackdelay          ;delay before returning
                     ;ENQ      stx     XMdelay
                     ;ENQ
                     ;ENQ2:    jsr     wait
                     ;ENQ
                     ;ENQ      ldx     XMdelay
                     ;ENQ      jne     2b
                     ;ENQ
                     ;ENQ1:
E67F 20 42E2                   jsr     wait              ;relinquish control
E682 4C C0E5                   jmp     toterm            ;go check for next char when you get back

E685 A9 00           nooutput:lda      #0
E687 85 04                     sta     outterm           ;nothing to do, turn term-write process off

E689 20 42E2                   jsr     wait              ;relinquish control
E68C 4C C0E5                   jmp     toterm            ;when you get it back, go check for more
                                                         ;work to do

                     ;*****************************************************************************

                     ;this routine decides what to do with the char depending on what the current
                     ;output state is set to
                                       ;state 0 - initial state, normal
                                       ;state 1 - <esc>, was previous char
                                       ;state 2 - <esc>'[' were 2 previous chars
                                       ;state 3 - <esc>'['[cnt]';' was previous output seq
                                       ;state 4 - don't interpret and map on output
                     ;states 2 and 3 are followed by an optional decimal count
                     ;states 1 through 3 are immediately followed by a command char
                     ;this whole sequence is used to generate cursor positioning
                     ;state0 is the normal state, in which chars are output to the screen
                     ;or if the char is an <esc>, state 1 is entered
E68F 85 7A           interp:   sta     savstat           ;save the current state
E691 D0 18                     bne     chkstat           ;if state non-zero, go resolve which state

E693 E0 20                     cpx     #SP               ;if char not a control, just output it
E695 B0 0C                     bcs     xnrm

E697 BC 00F8                   ldy     cntltab[x]        ;use char as first level index
E69A B9 01F9         dispatch:lda      termtab+1[y]      ;get high half of dispatch address
E69D 48                        pha                       ;push on stack for rts
E69E B9 00F9                   lda     termtab[y]        ;get low half
```

```
E6A1 48                   pha                     ;push on stack for rts
E6A2 60                   rts                     ;rts jumps to routine which handles this char

E6A3 E6 1E        xnrm:   inc    lcol
E6A5 20 0AE7              jsr    putout           ;output normal char

E6A8 A9 00                lda    #0
E6AA 60                   rts


E6AB C9 01        chkstat:cmp    #1               ;are we in state 1
E6AD D0 11                bne    1f               ;no, go check for states 2, 3 or 4

E6AF 8A                   txa                     ;yes, char -> acc
E6B0 38                   sec                     ;set the carry
E6B1 E9 40                sbc    #'@              ;subtract $40
E6B3 30 52                bmi    totermr0         ;not a legal char in state one,
                                                  ;go set state to 0 and wait for next char
E6B5 C9 20                cmp    #'_ - '@ + 1     ;if > '_', ignore entire sequence
E6B7 B0 4E                bcs    totermr0         ;go set state to 0 and wait for next char
E6B9 AA                   tax                     ;acc -> x-reg
E6BA BC 20F8              ldy    cltab[x]         ;get first level index for control table
E6BD 4C 9AE6              jmp    dispatch         ;save space by using state 0 dispatch routine

E6C0 C9 04        1:      cmp    #4               ;are we in state 4
E6C2 F0 3D                beq    s4               ;yes, go pass char through unmolested
                                                  ;no, must be in state 2
E6C4 8A                   txa                     ;char -> acc
E6C5 38                   sec                     ;set carry
E6C6 E9 30                sbc    #'0              ;if < '0', ignore entire sequence
E6C8 30 3D                bmi    totermr0         ;set to state 0, wait for next char
E6CA C9 10                cmp    #'@ - '0         ;if >= '@', then end of csi function
E6CC B0 25                bcs    excsi            ;go set up for dispatch
E6CE C9 0A                cmp    #10
E6D0 90 0A                bcc    2f               ;if <= '9', go add into parm[x]

E6D2 C9 0B                cmp    #'; - '0         ;is char a ';'
E6D4 D0 31                bne    totermr0         ;no, reset to state 0
E6D6 EE 9202              inc    parmindx
E6D9 A5 7A                lda    savstat          ;yes, wait for next char
E6DB 60                   rts

E6DC 8D BF02      2:      sta    temp             ;0 < acc < 10, save this val
E6DF AE 9202              ldx    parmindx         ;set up offset for parm1
E6E2 B5 16                lda    parm1[x]         ;get current parm1 or parm2
                                                  ;if state 2 -> parm1 = (parm1 * 10) + digit
                                                  ;if state 3 -> parm2 = (parm2 * 10) + digit
E6E4 0A                   asl                     ;multiply by 2
E6E5 0A                   asl                     ;multiply by 2
E6E6 18                   clc                     ;clear carry
E6E7 75 16                adc    parm1[x]         ;add in original parm1 val
E6E9 0A                   asl                     ;multiply by 2
E6EA 18                   clc                     ;clear carry
E6EB 6D BF02              adc    temp             ;add in next digit
E6EE 95 16                sta    parm1[x]         ;store away
E6F0 A5 7A                lda    savstat          ;stay in state 2 and wait for next char
```

```
E6F2 60                     rts

E6F3 C9 40          excsi:  cmp     #'p - '0        ;is char < 'p'
E6F5 B0 10                  bcs     totermr0        ;no, go set to state 0 and wait for next char
E6F7 38                     sec                     ;set carry
E6F8 E9 10                  sbc     #16             ;setup proper offset of index
E6FA AA                     tax                     ;acc -> x-reg
E6FB BC 40F8                ldy     csitab[x]       ;get first level index from csi func tab
E6FE 4C 9AE6                jmp     dispatch        ;save space by using state 0 dispatch routine

E701                s4:
E701 20 0AE7                jsr     putout
E704 A5 7A                  lda     savstat         ;remain in state 4 (-map)

E706 60                     rts

E707 A9 00          totermr0:lda    #0              ;jump here to set to state 0
E709 60                     rts                     ;jump here with new state in acc

                    ;*************************************************************************

                    ;this routine actually tries to output the char to the term
                    ;a wait is done if the transmit buffer is not empty in 2651-0

E70A                putout:
E70A A5 6A                  lda     map             ;if not in map mode, just put it out
E70C D0 30                  jne     nomap

E70E E0 0A                  cpx     #LF             ;is it a LF?
E710 D0 15                  jne     chkht           ;no, see if it is a HT

E712 A5 64                  lda     lfcr            ;should it be mapped into <lf><cr>?
E714 D0 28                  jne     nomap           ;no, just put it out

E716 A2 0D                  ldx     #CR
E718 A9 00                  lda     #0
E71A 85 1E                  sta     lcol
E71C 8D A002                sta     tabx

E71F 20 3EE7                jsr     addpar          ;put out CR

E722 A2 0A                  ldx     #LF
E724 4C 3EE7                jmp     addpar          ;now put out LF

E727                chkht:
E727 E0 09                  cpx     #HT             ;see if HT
E729 D0 13                  jne     nomap           ;no

E72B A5 68                  lda     tab             ;map tab into 8 spaces?
E72D F0 0F                  jeq     nomap           ;no, output as is

E72F A9 07                  lda     #7
E731 8D C002                sta     templ
E734 A2 20                  ldx     #SP
E736                0:
E736 20 3EE7                jsr     addpar
```

```
E739 CE C002             dec      templ
E73C D0 F8               jne      0b

E73E            nomap:
E73E A5 79      addpar:  lda      partype          ;if parity type none, don't mask char
E740 F0 04               beq      1f

E742 8A                  txa                       ;put char in acc
E743 29 7F               and      #$7F             ;make sure high order bit is 0
E745 AA                  tax                       ;restore char to x-reg

E746 20 6DE7    1:       jsr      putchar

E749 E0 0E               cpx      #CR + 1
E74B B0 12               bcs      2f               ;branch if >CR

E74D 8C C102             sty      temp4

E750 B4 6B               ldy      delays[x]        ;do delays on <BS> thru <CR>
E752 F0 08               beq      0f

E754 A2 00      1:       ldx      #NUL
E756 20 6DE7             jsr      putchar

E759 88                  dey
E75A D0 F8               bne      1b

E75C AC C102    0:       ldy      temp4

E75F A5 1E      2:       lda      lcol
E761 C9 50               cmp      #LINSIZ          ;see if end of line
E763 90 07               bcc      1f


E765 A9 00               lda      #0
E767 85 1E               sta      lcol             ;tidy counters
E769 8D A002             sta      tabx

E76C 60         1:       rts

                ;**********************************************************************

E76D AD 0110    putchar:lda       a10stat          ;find out status of term uart
E770 29 81               and      #(RTS ! trdy)    ;is transmit buffer empty
E772 C9 81               cmp      #(RTS ! trdy)    ;and flow control on
E774 D0 04               bne      2f               ;no, go wait

E776 8E 0010    1:       stx      a10data          ;yes, put out char
E779 60                  rts                       ;return

E77A 20 42[illegible]    2:       jsr      wait             ;if not ready, wait
E77D 4C 6DE7             jmp      putchar          ;then check status again

                ;**********************************************************************

E780 F0 07      reptcl:  beq      1f               ;output ! char even if y-reg contains 0
```

```

E782 20 0AE7       0:      jsr     putout
E785 88                    dey
E786 D0 FA         reptc0: bne     0b              ;output char n times, y-reg contains n

E788 60                    rts

E789 4C 0AE7       1:      jmp     putout

                   ;************************************************************************

E78C C0 00         adjy01: cpy     #0              ;adjust y, both 0 and 1 mean 1, no other changes
E78E D0 02                 bne     1f
E790 A0 01                 ldy     #1
E792 60            1:      rts

E793 C0 01         adjy10: cpy     #1              ;adjust y, all > 0 reduced by 1, 0=0
E795 10 02                 bpl     1f
E797 A0 01                 ldy     #1
E799 88            1:      dey
E79A 60                    rts

                   ;************************************************************************

                   ;this routine outputs error and status messages to the terminal
                   ;the acc and x-reg contain the address of the message to be output
                   ;the message is expected to be null terminated

E79B 20 A7E5       outmess:jsr     echoal
E79E 98                    tya
E79F 20 A7E5               jsr     echoal

E7A2 60                    rts






                   ;***************************************************************************

                   ;use this module when the host device can do out-of-band
                   ;flow control signalling for data going from device to the TIE.

E7A3               fcterm:
E7A3               ;QBF    lda     diag            ;if diag mode
E7A3               ;QBF    jne     2f              ;writing to a10cntr will upset loopback mode
E7A3               ;QBF
E7A3               ;QBF    lda     fctask
E7A3 C9 01                 cmp     #1              ;turn on flow?
E7A5 D0 05                 jne     0f

E7A7 A9 27                 lda     #$27            ;yes
E7A9 4C AEE7               jmp     1f

E7AC A9 07         0:      lda     #$07            ;no, uturn it off
E7AE 8D 0310       1:      sta     a10cntr
```

```

                   ;QBF2:
E7B1 A9 00                 lda     #0              ;turn off flow control task signal
E7B3 85 3A                 sta     fctask

E7B5 60                    rts                     ;and return

                   ;**********************************************************************

                   ;routines to receive a packet from the net

                   ; The control field looks like this:
                   ;
                   ;       MSB                                                          LSB
                   ;       ------------------------------------------------------------------
                   ;       |       |       |       |       |       |       |       |       |
                   ;       |   ACK NUMBER  | THIS PACKET'S | FLOW  |    PACKET             |
                   ;       |               | SEQUENCE NO.  | CNTL  |            TYPE       |
                   ;       |               |               |       |       |       |       |
                   ;       ------------------------------------------------------------------
                   ;
                   ;       PACKET TYPES:
                   ;       0 nop           1 data          2               3
                   ;       4 connect       5 disconnect    6               7 escape
                   ;
                   ;       ESCAPE (second control byte) PACKET TYPES:
                   ;       80 interrupt1   81 interrupt2   82 enquire      83 set tty
                   ;       84 get tty      85 current tty  86 sserv        87 report
                   ;


E7B6               rnumtab2:byte   $01,$02,$03,$00

E7BA               newrpnt:byte    0,2,4,0,1


                   ;this routine checks to see if a new packet has arrived from the net
                   ;if one has not, it checks to see if the watchdog timer has expired

E7BF AD 01C0       fmnet:  lda     ora6522         ;get the status of the net receiver
E7C2 29 07                 and     #7              ;low order 3 bits represent state of receiver buffers
E7C4 45 35                 eor     curbufs         ;check to see if it is a new packet
E7C6 D0 23                 bne     getpack         ;if it is, go suck it in

E7C8 AD 9002               lda     CONNstate       ;else, check to see if the TIE is
E7CB F0 18                 beq     frmretn         ;in the CONNECTED state, if it is not, go wait

E7CD A5 53                 lda     wdtimer         ;else, see if the watchdog timer has expired
E7CF D0 14                 bne     frmretn         ;if it has not, go wait

E7D1 C6 54                 dec     wdtimcnt
E7D3 F0 07                 beq     1f

E7D5 A9 3C                 lda     #60
E7D7 85 53                 sta     wdtimer
E7D9 4C E5E7               jmp     frmretn

E7DC A9 01         1:      lda     #1              ;else, set up a watchdog packet
E7DE 85 3D                 sta     inform
E7E0 85 06                 sta     netwrite

E7E2 EE E002               inc     wdcount         ;bump count of timeout watchdogs

E7E5 20 42E2       frmretn:jsr     wait            ;wait for a while
E7E8 4C BFE7               jmp     fmnet           ;then go try again
```

```

                    ;*********************************************************************

                    ;this routine decides whether the incoming packet
                    ;should really be accepted

E7EB 24 55          getpack:bit     rbufpnt         ;are we already pointing at it
E7ED D0 0A                  bne     3f

E7EF A6 55                  ldx    .rbufpnt         ;no, go through the table of pointers
E7F1 BC BAE7                ldy     newrpnt[x]      ;in a round robin fashion, until
E7F4 84 55                  sty     rbufpnt         ;we find the first one that matches
E7F6 4C EBE7                jmp     getpack

E7F9 A6 55          3:      ldx     rbufpnt         ;use the pointer to index the table
E7FB BD 0CEA                lda     rbufadh[x]      ;which contains the real address of the buffer
E7FE 85 37                  sta     currbufh        ;only need the high half
                                                    ;since buffers are on page boundaries

E800 A0 05                  ldy     #packtyp        ;look at control field in new packet
E802 B1 36                  lda     currbufl@[y]
E804 29 07                  and     #$07
E806 8D 9C02                sta     curptype        ;to get the packet type

E809 AD 9002                lda     CONNstate       ;if the TIE is connected or attempting
E80C D0 24                  bne     1f              ;a connection, accept a packet only from the
                                                    ;correct address

                    ;QBF    lda     monaddr         ;if in monitor mode
                    ;QBF    bne     legal           ;accept from any source
                    ;QBF
E80E AD 9C02                lda     curptype        ;else, make sure this is a CONN packet
E811 C9 04                  cmp     #CONN
E813 F0 03                  jne     xignore
E815 4C D4E8

E818 A0 01                  ldy     #dstaddr
E81A B1 36                  lda     currbufl@[y]    ;make sure the connect packet wasn't
E81C CD D8FF                cmp     SRCaddr
E81F F0 03                  jne     xignore         ;initiated by a "conn ffff"
E821 4C D4E8

E824 C8                     iny
E825 B1 36                  lda     currbufl@[y]
E827 CD D9FF                cmp     SRCaddr+1
E82A F0 03                  jne     xignore
E82C 4C D4E8

E82F 4C 47E8                jmp     legal

E832 A0 03          1:      ldy     #srcaddr        ;make sure that the packet is
E834 A5 59                  lda     distaddh        ;really from the distant TIE
E836 D1 36                  cmp     currbufl@[y]    ;party to the current connection
E838 F0 03                  jne     xignore
E83A 4C D4E8

```

```
E83D C8                     iny
E83E A5 58                  lda     distaddl        ;by checking both the high and low halfs
E840 D1 36                  cmp     currbufl@[y]    ;of the source address of this packet
E842 F0 03                  jne     xignore         ;if it doesn't match, release this buffer
E844 4C D4E8
                            .                       ;and go check for any other new packets

E847 A2 C6          legal:  ldx     #totrecd        ;increment total packets received
E849 20 BDED                jsr     inccount

E84C A0 05                  ldy     #packtyp        ;load y-reg with pointer to control field
E84E B1 36                  lda     currbufl@[y]    ;load the control field into the acc
E850 AA                     tax                     ;also store it for future use in the x-reg

E851 29 C0                  and     #$C0            ;mask off any other bits
E853 85 45                  sta     tnum            ;and store it away as current ack no.

E855 8A                     txa                     ;restore the control field
E856 29 08                  and     #$08            ;mask out all but flow control bit
E858 85 3E                  sta     drr             ;store it away

E85A AD 9C02                lda     curptype        ;look at packet type
E85D C9 04                  cmp     #CONN           ;if the packet is a connection request
E85F F0 14                  beq     0f              ;don't reset the watchdog timer

E861 C9 07                  cmp     #ESCAPE         ;or if this packet is an ENQ
E863 D0 08                  bne     1f

E865 A0 06                  ldy     #packtyp + 1
E867 B1 36                  lda     currbufl@[y]
E869 C9 82                  cmp     #ENQ
E86B F0 08                  beq     0f              ;don't reset the watchdog timer

E86D A9 3C          1:      lda     #60             ;else reset the watchdog timer to go off in
E86F 85 53                  sta     wdtimer         ;about 2 min
E871 A9 20                  lda     #32
E873 85 54                  sta     wdtimcnt

E875 AD 9C02        0:      lda     curptype        ;retrieve the packet type
E878 F0 5A                  beq     rellret         ;if it is zero(nop), go release the buffer

                    ;QBF    lda     monaddr         ;address monitor active?
                    ;QBF    beq     2f              ;no, skip
                    ;QBF
                    ;QBF    lda     curptype        ;if so, only look at the data packets
                    ;QBF    cmp     #1
                    ;QBF    bne     rellret         ;and ignore the rest
                    ;QBF    jmp     xconndat
                    ;QBF
                    ;QBF2:
E87A A9 01                  lda     #1              ;since it is not a nop, it must be acked
E87C 85 06                  sta     netwrite        ;make sure net xmitter is on
E87E 85 46                  sta     ackflag         ;make sure the ack is sent

E880 8A                     txa                     ;restore the control field
E881 29 30                  and     #$30            ;mask out all buf sequence number
```

```
E883 A4 44                ldy     rnum            ;get the expected sequence number
E885 D9 99EA              cmp     tabseq[y]       ;use it to get the correct bit representation of it
E888 D0 4A                bne     rellret         ;if packet does not have correct sequence no.
                                                  ;throw this packet away, the ack is still done

E88A B9 B6E7              lda     rnumtab2[y]     ;use current sequence number to get the
E88D 85 44                sta     rnum            ;next sequence number expected, and store

E88F AD 9C02              lda     curptype        ;if type is DATA
E892 C9 01                cmp     #DATA
E894 D0 02                jne     0f

E896 C6 49                dec     rbufcnt         ;then decrement count of free receive buffers

E898 A5 3F        0:      lda     pouts           ;check for a transmitted packet waiting for an ack
E89A F0 03                beq     1f              ;if not, continue

E89C 20 42E2              jsr     wait            ;else relinguish control to sched, so that
                                                  ;the net transmitter can see if this packet
                                                  ;contained the correct ack

E89F AD 9602      1:      lda     netstate        ;get the current network state
E8A2 0D 9C02              ora     curptype        ;or in the current packet type
E8A5 0A                   asl                     ;use this to determine the routine to handle
E8A6 AA                   tax                     ;this packet
E8A7 BD 96E9              lda     nettab+1[x]     ;get both the low and high halfs of the address
E8AA 48                   pha                     ;of the routine to execute
E8AB BD 95E9              lda     nettab[x]       ;by pushing them on the stack in the correct
E8AE 48                   pha                     ;order, a simple rts will then transfer
E8AF 60                   rts                     ;control to the selected routine

                  ;*******************************************************************************

                                                  ;network state = CONNECT, packet type = DISCON
E8B0 AD 9E02      xconndis:lda    discstat        ;check to see if a DISCON has already been sent
E8B3 C9 01                cmp     #SDISC          ;if it has not, que one up to be sent
E8B5 D0 12                bne     1f

E8B7 20 28EA              jsr     relrbufl        ;else release this buffer

E8BA A5 46        0:      lda     ackflag         ;and wait until the received packet has been
E8BC 05 47                ora     nosuccess
E8BE D0 03                jeq     sdiscon         ;acked, then go to routine which indicates
E8C0 4C 36ED

                                                  ;a disconnect has taken place
E8C3 20 42E2              jsr     wait            ;relinquish control to sched
E8C6 4C BAE8              jmp     0b              ;keep checking

E8C9 A9 02        1:      lda     #RDISC          ;set state to indicate a DISCON has been
E8CB 8D 9E02              sta     discstat        ;received, but not sent

E8CE A9 01                lda     #1              ;set flag to send a DISC packet
E8D0 85 3[illegible]              sta     sendisc
E8D2 85 [illegible]               sta     netwrite        ;and to activate the netwrite process

```

```
                   ;QBF    lda     #0              ;turn of the qbf if its on
                   ;QBF    sta     doqbf
                   ;QBF    sta     echo.off        ;turn the echo back on
                   ;QBF
E8D4               rellret:                        ;share some code
E8D4 20 28EA       xignore:jsr     relrbufl        ;release this buffer

E8D7 20 42E2               jsr     wait            ;relinquish control to sched
E8DA 4C BFE7               jmp     fmnet           ;go check for new packets

                   ;***********************************************************************

                                                   ;network state = CONNECT, packet type = DATA
E8DD A6 2A         xconndat:ldx    recinp          ;get the input pointer into the packet que
E8DF A5 55                 lda     rbufpnt         ;get the pointer to the buffer that contains this packet
E8E1 95 2D                 sta     recque[x]       ;put it on the que for the term-tranmsitter
E8E3 BC EAE2               ldy     nxtindx[x]      ;use current input pointer
E8E6 84 2A                 sty     recinp          ;to get next input pointer and store
E8E8 E6 2C                 inc     reccnt          ;increment count of things in que

E8EA 05 35                 ora     curbufs         ;curbufs contains bit pointers to all active
E8EC 85 35                 sta     curbufs         ;buffers, it is used to determine if a packet
                                                   ;has already been handled

E8EE A9 01                 lda     #1              ;make sure the terminal-transmitter process
E8F0 85 04                 sta     outterm         ;gets scheduled

E8F2 A2 CC                 ldx     #datarecd       ;increment total data packets received
E8F4 20 BDED               jsr     inccount

E8F7 20 42E2               jsr     wait            ;relinquish control to sched
E8FA 4C BFE7               jmp     fmnet           ;when you get it back, go check for new packet

                   ;***********************************************************************

                                                   ;network state = IDLE, packet type = CONN
E8FD A9 10         xidlecon:lda    #RCON           ;set network state -> RCON
E8FF 8D 9602               sta     netstate

E902 A9 04                 lda     #CONN           ;and que up a CONN
E904 20 46E4               jsr     quepack         ;for the net-transmitter

E907 A0 03                 ldy     #srcaddr        ;find out the source address
E909 B1 36                 lda     currbufl@[y]    ;of this packet
E90B 85 59                 sta     distaddh        ;and save it for future comparisons
E90D 8D 0102               sta     trbuf0 + dstaddr        ;of source address in incoming packets
E910 8D 0103               sta     trbufl + dstaddr        ;also make it the destination address
E913 8D 0104               sta     trbuf2 + dstaddr        ;of all outgoing packets

E916 C8                    iny                     ;since it is a 16-bit address
E917 B1 36                 lda     currbufl@[y]    ;both the low and high halfs must
E919 85 58                 sta     distaddl        ;be done
E91B 8D 0202               sta     trbuf0 + dstaddr + 1
E91E 8D 0203               sta     trbufl + dstaddr + 1
E921 8D 0204               sta     trbuf2 + dstaddr + 1

```

```
E924 A9 01                 lda     #BUSY           ;set this state to indicate that no other
E926 8D 9002               sta     CONNstate       ;connections can be initiatied either
                                                   ;externally or internally
E929 4C D4E8               jmp     rellret         ;go release this buffer

                  ;*********************************************************************

                                                   ;network state = SCON, packet type = CONN
E92C A9 18        xsconcon:lda     #CONNECT        ;set the network state -> CONNECT
E92E 8D 9602               sta     netstate        ;since the connection protocol was completed

E931 A9 00                 lda     #0
E933 85 0F                 sta     tin.brk         ;make sure this is turned off, since it
E935 20 A5ED               jsr     conmess         ;inform user that connection was successful
                                                   ;inidicates thata data buffer is ready to send


E938 4C D4E8               jmp     rellret         ;go release this buffer

                  ;*********************************************************************

                                                   ;network state = CONNECT, packet type = ESCAPE
E93B A0 06        xconnesc:ldy     #packtyp + 1    ;find out the real packet type
E93D B1 36                 lda     currbufl@[y]    ;which is contained in the next location

E93F 29 7F                 and     #$7F            ;get rid of high order bit
E941 0A                    asl                     ;multiply by 2 to use as index into table
E942 AA                    tax                     ;of address of routines to handle each type

E943 BD D6E9               lda     esctab+1[x]
E946 48                    pha
E947 BD D5E9               lda     esctab[x]
E94A 48                    pha
E94B 60                    rts

                  ;*********************************************************************

                                                   ;network state = CONNECT, packet type = INTR1
E94C 20 E3E9      xintl:   jsr     clrtrbfs        ;go release all the transmit buffers

E94F A9 81                 lda     #INTR2          ;que up a response to the INTR1
E951 20 46E4               jsr     quepack         ;share some code

                  ;*********************************************************************

                                                   ;network state = CONNECT, packet type = INTR2
E954 20 75E9      xint2:   jsr     relall          ;go release all the receiver buffers

E957 20 42E2               jsr     wait            ;relinquish control to sched
E95A 4C BFE7               jmp     fmnet           ;go check for new packets

                  ;*********************************************************************

E95D A0 07        xstty:   ldy     #packtyp + 2    ;point at echo parameter

E95F B1 36                 lda     currbufl@[y]
```

```
E961 85 65                sta     echo.off         ;set echo parameter

E963 C8                   iny                      ;point at raw/cooked parameter
E964 B1 36                lda     currbufl@[y]
E966 85 62                sta     rawcook          ;set raw/cooked parameter
E968 85 7B                sta     instatl

E96A 4C D4E8              jmp     rellret          ;go release the buffer

                  ;***********************************************************************

E96D A9 85        xgtty:  lda     #CTTY            ;send a current tty packet
E96F 20 46E4              jsr     quepack

E972 4C D4E8              jmp     rellret

                  ;***********************************************************************


                                                   ;this routine releases all the receiver buffers
                                                   ;making them available to the hardware
E975 8D 021C      relall: sta     clrrb0           ;by accessing these address the buffer full
E978 8D 041C              sta     clrrbl           ;flags associated with each buffer are
E97B 8D 061C              sta     clrrb2           ;cleared

E97E A9 00                lda     #0               ;make sure the que of full receiver buffers
E980 85 2A                sta     recinp           ;is re-initialized
E982 85 2B                sta     recoutp
E984 85 4A                sta     currcnt
E986 85 2C                sta     reccnt
E988 85 35                sta     curbufs          ;zero the bit pointers

E98A A9 02                lda     #2               ;reset the count of available buffers for
E98C 85 49                sta     rbufcnt          ;data packets

E98E A9 01                lda     #1
E990 85 3D                sta     inform           ;the state of the other side
E992 85 06                sta     netwrite

E994 60                   rts

                  ;***********************************************************************

E995              nettab: addr    xignore-1,xignore-1,xignore-1,xignore-1         ;0  2  4  6
E99D                      addr    xidlecon-1,xignore-1,xignore-1,xignore-1        ;8  10 12 14
E9A5                      addr    xignore-1,xignore-1,xignore-1,xignore-1         ;16 18 20 22
E9AD                      addr    xsconcon-1,xignore-1,xignore-1,xignore-1        ;24 26 28 30
E9B5                      addr    xignore-1,xignore-1,xignore-1,xignore-1         ;32 34 36 38
E9BD                      addr    xignore-1,xignore-1,xignore-1,xignore-1         ;40 42 44 46
E9C5                      addr    xignore-1,xconndat-1,xignore-1,xignore-1        ;48 50 52 54
E9CD                      addr    xignore-1,xconndis-1,xignore-1,xconnesc-1       ;56 58 60 62

E9D5              esctab: addr    xintl-1,xint2-1,xignore-1,xstty-1       ;0  2  4  6
E9DD                      addr    xgtty-1,xignore-1,xignore-1             ;8 10 12

                  ;***********************************************************************
```

```

                                                 ;this routine clears out all the transmit
                                                 ;buffers, and releases them for use
E9E3 A9 00          clrtrbfs:lda    #0           ;make sure that input and output routines
E9E5 85 57                   sta    stopinp      ;are turned back on
E9E7 85 56                   sta    stopoutp
E9E9 8D 9D02                 sta    goreleas

E9EC 85 1F                   sta    queent       ;clean up the que of things to do
E9EE 85 28                   sta    queinp
E9F0 85 29                   sta    queoutp

E9F2 85 0E                   sta    tin.cnt      ;zero the count of current chars
E9F4 85 0F                   sta    tin.brk      ;indicate no break condition

E9F6 85 0A                   sta    tran.used    ;zero the count of used transmit buffers

E9F8 A9 01                   lda    #1           ;turn on flow from term
E9FA 85 3A                   sta    fctask
E9FC 85 04                   sta    outterm

E9FE A9 06                   lda    #trstart     ;reset the pointer into the current
EA00 85 0D                   sta    tin.p        ;transmit buffer being filled



EA02 A5 11                   lda    tout.buf + 1 ;make sure the current pointers to the current
EA04 85 0C                   sta    tin.buf + 1  ;input buffer and next buffer to be transmitted
                                                 ;are the same
EA06 60                      rts                 ;done

                    ;***********************************************************************



                    ;this routine releases a reciever buf back to the net
                    ;it has 2 entry points

EA07                reltab: byte    0,2,4,0,6
EA0C                rbufadh:byte    0,5,6,0,7

EA11 A5 4B          relrbuf:lda     currbufx     ;put bit rep of buffer into acc
EA13 45 35                   eor    curbufs      ;xor it with bit rep of current used bufs
EA15 85 35                   sta    curbufs      ;this turns off bit representing buf in ques

EA17 A5 49                   lda    rbufcnt      ;get cur count of free receiv bufs
EA19 D0 06                   bne    1f           ;if non-zero other end of conn ok

EA1B A9 01                   lda    #1           ;otherwise force the net to send
EA1D 85 3D                   sta    inform       ;a watchdog packet, which will enable
EA1F 85 C6                   sta    netwrite     ;the other side to send since the RR
                                                 ;flag has been turned off

EA21 E6 49          1:       inc    rbufcnt      ;bump up count of free rec bufs
EA23 A6 4B                   ldx    currbufx     ;get bit pointer to buffer to be released
EA25 4C [illegible]A         jmp    0f           ;go release

```

```
EA28 A6 55       relrbufl:ldx    rbufpnt         ;get bit pointer to buffer to be released
EA2A BD 07EA     0:       lda    reltab[x]       ;get address that releases this buf
EA2D 85 38                sta    relbufl         ;store it on page 0
EA2F A0 00                ldy    #0              ;no offset necessary
EA31 91 38                sta    relbufl@[y]     ;just accessing this mem loc releases this buf
EA33 60                   rts
```

```
EA34 A9 00          gonop:  lda     #0
EA36 85 46                  sta     ackflag
EA38 A4 44                  ldy     rnum
EA3A 19 9DEA                ora     taback[y]
EA3D A4 49                  ldy     rbufcnt
EA3F F0 02                  jeq     1f

EA41 09 08                  ora     #RRb

EA43 A0 05          1:      ldy     #HEADSIZ
EA45 8C 0004                sty     trbuf2 + hardcnt
EA48 8D 0504                sta     trbuf2 + packtyp
EA4B A9 A4                  lda     #trbuf2pt
EA4D 8D 00C0                sta     orb6522

EA50 A2 00                  ldx     #0
EA52 86 40                  stx     colcnt

EA54 A9 01                  lda     #1
EA56 85 47                  sta     nosuccess

EA58 A6 40          0:      ldx     colcnt
EA5A BD DAFF                lda     backoffl[x]
EA5D 8D 04C0                sta     t1l6522
EA60 BD EAFF                lda     backoffh[x]
EA63 8D 05C0                sta     t1h6522

EA66 20 42E2        8:      jsr     wait
EA69 AD 01C0                lda     ora6522
EA6C 10 1D                  bpl     1f

EA6E A5 40                  lda     colcnt
EA70 C9 0F                  cmp     #COLMAX
EA72 D0 03                  jeq     broken
EA74 4C 1BED

EA77 E6 40                  inc     colcnt

EA79 A2 DC                  ldx     #totcol
EA7B 20 BDED                jsr     inccount

EA7E A5 40                  lda     colcnt
EA80 CD DF02                cmp     maxcol
EA83 90 03                  bcc     9f

EA85 8D DF02                sta     maxcol

EA88 4C 58EA        9:      jmp     0b

EA8B 29 40          1:      and     #success
EA8D F0 D7                  jeq     8b

EA8F A9 00                  lda     #0
EA91 85 47                  sta     nosuccess

EA93 A2 E3                  ldx     #tottrns
```

```
EA95 20 BDED           jsr     inccount

EA98 60                rts

```

```
                   ;routines to put a packet out on the net

                   ;
                   ;       MSB                                                    LSB
                   ;       ---------------------------------------------------------------
                   ;       |       |        |       |       |       |       |       |       |
                   ;       |   ACK NUMBER   |  THIS PACKET'S  | FLOW  |     PACKET          |
                   ;       |                |  SEQUENCE NO.   | CNTL  |            TYPE     |
                   ;       |                |                 |       |       |       |     |
                   ;       ---------------------------------------------------------------
                   ;
                   ;       PACKET TYPES:
                   ;       0 nop          1 data          2               3
                   ;       4 connect      5 disconnect    6               7 escape
                   ;
                   ;       ESCAPE (second control byte) PACKET TYPES:
                   ;       80 interrupt1   81 interrupt2   82 enquire      83 set tty
                   ;       84 get tty      85 current tty  86 sserv        87 report
                   ;

EA99               tabseq: byte    $10,$20,$30,$00
EA9D               taback: byte    $00,$40,$80,$C0
EAA1               tabackx:byte    $40,$80,$C0,$00

EAA5               repcnt: byte    0,7,7,7,2,3,7,7,7,7,7,7,7,7,7
                   ; number of retries for "disc" changed from 2 to 3 - 790902

EAB4               hostsiz:byte    2,4,8,12,16,20,24,28,32,32,32,32,32,32,32,32

EAC4               ackwait:byte    8,8,12,16,24,40,40,40


EACC               tonet:
EACC               ;QBF    lda     monaddr         ;if address-monitor mode
EACC               ;QBF    jne     mnwait          ;don't write to the net
EACC               ;QBF
EACC A5 3C                 lda     sendisc         ;should a DISC packet be sent?
EACE F0 08                 beq     1f              ;no

EAD0 A5 1F                 lda     quecnt          ;if so, is anything on the que?
EAD2 F0 25                 beq     forcDISC        ;if not, then send the DISC now

EAD4 A5 51                 lda     distimer        ;if so, has the timer waiting for the que
EAD6 F0 21                 beq     forcDISC        ;to empty ticked down?

EAD8 A5 1F         1:      lda     quecnt          ;if there is anything on the que
EADA D0 33                 bne     fmque           ;to do, go do it

EADC A5 3D         0:      lda     inform          ;check on connection integrity?
EADE D0 24                 bne     forcENQ         ;yes, do it

EAE0 A5 46                 lda     ackflag         ;check to see if last packet received
EAE2 F0 07                 jeq     1f              ;has been acked ... if so, continue

EAE4 A9 00                 lda     #0              ;if not, do it with a nop packet
EAE6 85 4F                 sta     curtyp
```

```
EAE8 4C 3FEB              jmp     subnop

EAEB A5 1F        1:      lda     quecnt          ;if something on the que
EAED D0 04                bne     mnwait          ;don't turn off netwrite

EAEF A9 00                lda      #0
EAF1 85 06                sta     netwrite        ;else turn off this process, nothing to do

EAF3 20 42E2      mnwait: jsr     wait            ;relinquish control to the scheduler
EAF6 4C CCEA              jmp     tonet           ;when we get it back, go check again

                  ;************************************************************************

EAF9 A9 05        forcDISC:lda    #DISCON         ;force DISC to be the next packet sent
EAFB 85 4F                sta     curtyp

EAFD A9 00                lda     #0              ;reset the send DISC flag
EAFF 85 3C                sta     sendisc

EB01 4C 35EB              jmp     submit

                  ;************************************************************************

EB04 A9 82        forcENQ:lda     #ENQ            ;send ENQ packet to check on the connection
EB06 85 4F                sta     curtyp

EB08 A9 00                lda     #0
EB0A 85 3D                sta     inform          ;reset flag to send ENQ

EB0C 4C 35EB              jmp     submit          ;and submit the packet

                  ;*************************************************************************

EB0F A6 29        fmque:  ldx     queoutp         ;get pointer to next task
EB11 B5 20                lda     que[x]          ;get the packet type to be sent
EB13 85 4F                sta     curtyp          ;store it away

EB15 AE 9602              ldx     netstate        ;see if we are in the CONNECT state
EB18 E0 18                cpx     #CONNECT
EB1A F0 0E                beq     1f              ;if we are go see if the packet can be sent

EB1C C9 04                cmp     #CONN           ;else, if the packet to be sent is a CONN
EB1E F0 12                jeq     2f              ;go ahead and do it

EB20 C9 87                cmp     #REPT           ;or if a REPT packet
EB22 F0 0E                jeq     2f              ;go ahead and send it
                                                  ;(in case connection attempt failed)

EB24 20 B3ED              jsr     decque          ;else just take it off the que, don't send

EB27 4C F3EA              jmp     mnwait          ;and then go wait

EB2A C9 01        1:      cmp     #DATA           ;if the packet isn't data go ahead
EB2C D0 04                jne     2f

EB2E A5 3E                lda     drr             ;else if the other side has room for this
```

```
EB30 F0 AA                  beq     0b                ;data packet, go ahead

EB32 20 B3ED        2:      jsr     decque            ;not a nop, take if off the que

EB35 A4 42          submit: ldy     cursent           ;get the last sequence number sent
EB37 B9 A1EA                lda     tabackx[y]        ;get correct bit representation of expected ack
EB3A 85 43                  sta     cursub            ;save it for comparison with incoming acks
EB3C B9 99EA                lda     tabseq[y]         ;use it to get the next sequence number to send

EB3F A4 44          subnop: ldy     rnum              ;get the sequence number of the last packet recevied
EB41 19 9DEA                ora     taback[y]         ;OR in the ack #

EB44 A4 49                  ldy     rbufcnt           ;find out the number of free receive buffers
EB46 F0 02                  beq     2f                ;for data, if non-zero
EB48 09 08                  ora     #RRb               ;tell the other guy he can send data

EB4A A6 4F          2:      ldx     curtyp            ;get the type of packet being sent
EB4C 86 50                  stx     realtyp           ;save it away, in case it needs an ESCAPE
EB4E 10 60                  bpl     3f                ;go do a normal packet

EB50 A0 06                  ldy     #HEADSIZ + 1              ;else it needs an ESCAPE
EB52 8C 0004                sty     trbuf2 + hardcnt          ;store the count in the buffer reserved

EB55 E0 87                  cpx     #REPT             ;is this the report datagram
EB57 D0 36                  bne     5f                ;no

EB59 A0 2A                  ldy     #RPTSIZ
EB5B 8C 0004                sty     trbuf2 + hardcnt          ;and set up the hard count

EB5E A6 41                  ldx     tcnt              ;if this is not the first tran attempt
EB60 D0 40                  bne     4f                ;don't need (or want) to form the report again

EB62 A4 59                  ldy     distaddh          ;save the distant address
EB64 8C 2904                sty     trbuf2 + RPTSIZ - 1       ;as the other party to this connection
EB67 A4 58                  ldy     distaddl
EB69 8C 2A04                sty     trbuf2 + RPTSIZ

EB6C A0 01                  ldy     #rptaddr > 8              ;and make the report address
EB6E 8C 0104                sty     trbuf2 + dstaddr          ;the new distant address
EB71 84 59                  sty     distaddh
EB73 A0 00                  ldy     #rptaddr
EB75 8C 0204                sty     trbuf2 + dstaddr + 1
EB78 84 58                  sty     distaddl

EB7A 48                     pha                       ;save the acc ( control field)

EB7B A0 07                  ldy     #7
EB7D A2 00                  ldx     #0
EB7F BD C3C2        0:      lda     tottrns[x]        ;move all the counters
EB82 99 0004                sta     trbuf2[y]         ;and the elapsed time into the buffer

EB85 E8                     inx
EB86 C8                     iny
EB87 E0 22                  cpx     #RPTSIZ - 8
EB89 D0 F4                  bne     0b

```

```
EB8B 68                 pla                             ;restore the acc

EB8C 4C A2EB            jmp     4f

EB8F E0 85      5:      cpx     #CTTY                   ;is it a current tty packet
EB91 D0 0F              bne     4f                      ;no, continue

EB93 A4 65              ldy     echo.off                ;else set up echo
EB95 8C 0704            sty     trbuf2 + packtyp + 2    ;and
EB98 A4 62              ldy     rawcook                 ;raw/cooked
EB9A 8C 0804            sty     trbuf2 + packtyp + 3    ;paramaters

EB9D A0 08              ldy     #HEADSIZ + 3            ;and adjust for the correct count
EB9F 8C 0004            sty     trbuf2 + hardcnt
                                                        ;for control packets
EBA2 09 07      4:      ora     #ESCAPE                 ;set the type to be ESCAPE
EBA4 A6 50              ldx     realtyp                 ;get the real type of packet
EBA6 8E 0604            stx     trbuf2 + packtyp + 1    ;and stick it in location packtyp + 1:
EBA9 A2 07              ldx     #ESCAPE                 ;make the real type ESCAPE, for
EBAB 86 50              stx     realtyp                 ;purposes of table lookups
EBAD 4C BBEB            jmp     5f                      ;skip uneccessary code

EBB0 05 50      3:      ora     realtyp         ;if it was a normal packet, set up the type
EBB2 E0 01              cpx     #DATA           ;see if it was of type DATA
EBB4 F0 0D              beq     1f              ;if so, skip this code

EBB6 A0 05              ldy     #HEADSIZ                ;set up size of normal control packet
EBB8 8C 0004            sty     trbuf2 + hardcnt        ;store it in the control buffer

EBBB 8D 0504    5:      sta     trbuf2 + packtyp        ;set the control field in the control
EBBE A9 A4              lda     #trbuf2pt               ;packet, and get the pointer to
EBC0 4C C9EB            jmp     2f                      ;buffer, skip some code

EBC3 A0 05      1:      ldy     #packtyp        ;DATA type, set up the control field
EBC5 91 10              sta     tout.buf@[y]    ;in the current data buffer to be transmitted

EBC7 A5 48              lda     curtrbuf        ;get the current pointer to the buffer
EBC9 8D 00C0    2:      sta     orb6522         ;store in the hardware register

EBCC A9 01              lda     #1
EBCE 85 47              sta     nosuccess       ;indicate no success xmit yet

EBD0 A9 00              lda     #0              ;set the collision count for this packet
EBD2 85 40              sta     colcnt          ;to 0
EBD4 85 46              sta     ackflag         ;turn off the acknowledge needed flag

EBD6 A6 40      settim: ldx     colcnt
EBD8 BD DAFF            lda     backoffl[x]     ;correct backoff time for this packet
EBDB 8D 04C0            sta     tl16522
EBDE BD EAFF            lda     backoffh[x]     ;store it in the hardware timer, when it
EBE1 8D 05C0            sta     tlh6522         ;counts down, the packet will be submitted for
                                                ;transmission
EBE4 20 42E2    notsent:jsr     wait            ;wait awhile

EBE7 AD 01C0            lda     ora6522         ;get the status of the transmission
EBEA 10 1D              bpl     1f              ;if the high order bit was set, a collision
```

```
                                                         ;occurred, else go check for success
EBEC A5 40                  lda      colcnt              ;get the current collision count
EBEE C9 0F                  cmp      #COLMAX             ;see if we have exceeded the maximun number
EBF0 D0 03                  jeq      broken              ;if we have, go punt
EBF2 4C 1BED

EBF5 E6 40                  inc      colcnt              ;else increment the collison count

EBF7 A2 DC                  ldx      #totcol             ;get address of total collision count
EBF9 20 BDED                jsr      inccount            ;and increment it

EBFC A5 40                  lda      colcnt              ;have we exceeded the record max on a given packet?
EBFE CD DF02                cmp      maxcol
EC01 90 03                  bcc      9f
EC03 8D DF02                sta      maxcol              ;if yes, this is the new record

EC06 4C D6EB        9:      jmp      settim              ;go resubmit the identical packet
                                                         ;to the hardware

                    ;***********************************************************************

EC09 29 40          1:      and      #success            ;check for successful transmission
EC0B F0 D7                  beq      notsent             ;if not go wait some more

EC0D A9 00                  lda      #0
EC0F 85 47                  sta      nosuccess           ;indicate packet has been transmitted

EC11 A2 C3                  ldx      #tottrns            ;increment total successful xmissions
EC13 20 BDED                jsr      inccount

                                                         ;SUCCESSFUL TRANSMISSION
EC16 A5 50                  lda      realtyp             ;see if the real type was a NOP
EC18 F0 72                  beq      3f                  ;if so, don't bother to wait for an ack

EC1A C9 01                  cmp      #DATA               ;if this is a data packet
EC1C D0 05                  bne      1f

EC1E A2 C9                  ldx      #datatrns           ;increment the total data packets trans
EC20 20 BDED                jsr      inccount

EC23 A9 01          1:      lda      #1                  ;else indicate, there is a packet waiting
EC25 85 3F                  sta      pouts               ;for an ack from the other guy

                    ;** 790807 - Mike, I have changed this to reuse the acktimer table
                    ;       if we allow more than 7 tries.  RJC
                    ;       lda      tcnt                ;get the current count of transmissions of
                    ;       and      #7                  ;this packet, mask to 3 bits,
                    ;       tax
EC27 A6 41                  ldx      tcnt
EC29 BD C4EA                lda      ackwait[x]          ;use it to get the time to
EC2C 85 52                  sta      acktimer            ;wait after the current transmission
                                                         ;this time increases, with the number of retries
EC2E 20 42E2        2:      jsr      wait                ;wait a while

EC31 A5 45                  lda      tnum                ;see if this packet has been acked yet
EC33 C5 43                  cmp      cursub
```

```
EC35 F0 55                  beq     3f              ;if it has, go do SUCCESSFUL ACKNOWLEDGEMENT

EC37 A5 46                  lda     ackflag         ;see if we have to send a nop ack
EC39 F0 03                  jeq     0f              ;for a cross in the mail

EC3B 20 34EA                jsr     gonop

EC3E A5 52          0:      lda     acktimer        ;else see if the ack timer has gone off yet
EC40 D0 EC                  bne     2b              ;if it hasn't, go wait some more

EC42 A5 4F                  lda     curtyp          ;get packet type
EC44 10 05                  bpl     1f              ;if not ESCAPE, skip

EC46 29 7F                  and     #$7F            ;strip off ESCAPE bit
EC48 18                     clc
EC49 69 07                  adc     #7              ;and add 7 to index into repcnt table

EC4B A8             1:      tay

EC4C C0 04                  cpy     #CONN           ;if it is a CONN, do code for rotary
EC4E D0 1E                  bne     0f              ;else go handle other types

EC50 AD 9A02                lda     hostconn        ;see if we are trying to connect to a host
EC53 F0 19                  beq     0f              ;with a rotary, if not go handle like other
                                                    ;types
EC55 EE 9B02                inc     conncnt         ;else, increment current count of connection
EC58 CD 9B02                cmp     conncnt         ;attempts, if we are at the end
EC5B D0 03                  jeq     broken          ;of the rotary, punt
EC5D 4C 1BED

EC60 E6 58                  inc     distaddl                ;else inc the address we are attempting
EC62 EE 0202                inc     trbuf0 + dstaddr + 1    ;the connection to, and store it
EC65 EE 0203                inc     trbuf1 + dstaddr + 1    ;away, in all of the transmit buffers
EC68 EE 0204                inc     trbuf2 + dstaddr + 1

EC6B 4C 35EB                jmp     submit          ;go resubmit the CONN packet

EC6E A5 41          0:      lda     tcnt            ;else get the transmission count of this packet
EC70 D9 A5EA                cmp     repcnt[y]       ;use the type, to indicate the maximum number
EC73 D0 03                  jeq     broken          ;of retransmissions, if exceeded punt
EC75 4C 1BED

EC78 E6 41                  inc     tcnt            ;else increment the transmission count

EC7A A2 D7                  ldx     #totlost
EC7C 20 BDED                jsr     inccount        ;go increment total count of lost packets

EC7F A5 41                  lda     tcnt
EC81 CD DA02                cmp     maxlost         ;is this a record retrans count for a given packet?
EC84 90 03                  bcc     9f
EC86 8D DA02                sta     maxlost         ;if so, set it

EC89 4C 35EB        9:      jmp     submit          ;and go update the control field before
                                                    ;retransmission

                    ;**********************************************************************
```

```

                                                      ;SUCCESSFUL TRANSMIT AND ACKNOWLEDGEMENT

ECBC A2 00          3:     ldx      #0                ;the packet has been acked
ECBE 86 3F                 stx      pouts             ;indicate no packet waiting for ack
EC90 86 41                 stx      tcnt              ;zero transmission count
EC92 86 52                 stx      acktimer          ;no longer needed, turn it off

EC94 AD DA02               lda      maxlost           ;if the new maxlost
EC97 CD DB02               cmp      lastlost          ;beats the old one
EC9A 90 03                 bcc      1f

EC9C 8D DB02               sta      lastlost          ;save it

EC9F A5 4F          1:     lda      curtyp           ;if it was a NOP, we are done
ECA1 F0 72                 beq      done

ECA3 A6 42                 ldx      cursent           ;else increment the sequence number
ECA5 BC B6E7               ldy      rnumtab2[x]       ;which reflects the last packet that
ECA8 84 42                 sty      cursent           ;has been successfully sent

ECAA C9 01                 cmp      #DATA             ;see if it was a DATA packet
ECAC D0 17                 bne      1f                ;if not skip code

ECAE A5 48                 lda      curtrbuf          ;else, get the pointer to the DATA buffer
ECB0 49 01                 eor      #1                ;which is currently to be transmitted
ECB2 85 48                 sta      curtrbuf          ;change it to point at the other one

ECB4 A5 11                 lda      tout.buf + 1      ;update the address of the buffer in the
ECB6 49 01                 eor      #1                ;same manner
ECB8 85 11                 sta      tout.buf + 1

ECBA A9 01                 lda      #1                ;make sure flow from term
ECBC 85 3A                 sta      fctask            ;is turned on
ECBE 85 04                 sta      outterm

ECC0 C6 0A                 dec      tran.used         ;decrement count of used DATA buffers
ECC2 4C 15ED               jmp      done              ;go finish up

ECC5 C9 87          1:     cmp      #REPT             ;if this was a report datagram
ECC7 D0 03                 jeq      init              ;then reset the tie
ECC9 4C 00E1

ECCC C9 05                 cmp      #DISCON           ;see if the packet was a DISCON
ECCE D0 12                 bne      1f                ;if not, skip some code

ECD0 AD 9E02               lda      discstat          ;else, get the current state of the disconnect
ECD3 C9 02                 cmp      #RDISC            ;sequence, if a DISCON has already been
ECD5 D0 03                 jeq      rdiscon           ;received, go finish
ECD7 4C 63ED

ECDA A9 01                 lda      #SDISC            ;else set the state to sent DISCON
ECDC 8D 9E02               sta      discstat          ;waiting to receive one

ECDF 4C 15ED               jmp      done              ;go clean up

```

```
ECE2 C9 04       1:       cmp     #CONN          ;see if the packet was a CONN
ECE4 D0 2F                bne     done           ;if it wasn't go clean up

ECE6 AD 9602              lda     netstate       ;else get the state of the connect sequence
ECE9 D0 1A                bne     1f             ;if the state is not idle go check some more

ECEB A9 08                lda     #SCON          ;else set the state to sent a CONN
ECED 8D 9602              sta     netstate       ;waiting to receive one

ECF0 A9 FF                lda     #255           ;set timer to go off in 16.6 sec
ECF2 85 52                sta     acktimer

ECF4 20 42E2     0:       jsr     wait

ECF7 AD 9602              lda     netstate       ;and wait for state to become CONNECTed
ECFA C9 18                cmp     #CONNECT
ECFC F0 17                beq     done           ;and then continue

ECFE A5 52                lda     acktimer       ;if we're not connected at the end of 16.6 sec
ED00 D0 F2                bne     0b

ED02 4C 1BED              jmp     broken         ;then inform the user and reset the tie

ED05 C9 10       1:       cmp     #RCON          ;see if a CONN has been received yet
ED07 D0 0C                bne     done           ;if not go clean up

ED09 A9 18                lda     #CONNECT       ;else, set the state to connected
ED0B 8D 9602              sta     netstate       ;ready to accept data

ED0E A9 00                lda     #0             ;make sure this is reset
ED10 85 0F                sta     tin.brk

ED12 20 A5ED              jsr     conmess

ED15 20 42E2     done:    jsr     wait           ;relinquish control for awhile
ED18 4C CCEA              jmp     tonet          ;when you get it back, go check for anything
                                                 ;else to do

                 ;************************************************************************

                                                 ;come here if too many collisions, or too
ED1B A5 4F       broken:  lda     curtyp        ;many retransmissions of a pakets
ED1D C9 87                cmp     #REPT          ;if this was a report datagram
ED1F D0 03                jeq     init           ;then just reset and don't worry about it
ED21 4C 00E1

ED24 C9 04                cmp     #CONN          ;see if the packet type was a CONN
ED26 D0 07                bne     1f             ;if not, skip code

ED28 A0 E5                ldy     #noconn        ;else, indicate that the connection
ED2A A9 FD                lda     #noconn > 8    ;was not completed
ED2C 4C 6AED              jmp     3f

ED2F A0 D2       1:       ldy     #conbroke      ;indicate that the connection was broken
ED31 A9 FD                lda     #conbroke > 8

```

```
ED33 4C 6AED                jmp     3f


ED36 20 8DED        sdiscon:jsr     waitnet          ;come here if this TIE initiated the discon
                                                     ;sequence, and wait until the output from
                                                     ;from the network is finished
ED39 A0 C4                  ldy     #messdisc        ;inform the user that the disconnect has
ED3B A9 FD                  lda     #messdisc > 8    ;taken place

ED3D 20 9BE7                jsr     outmess          ;can't share the code now

ED40 20 DDED                jsr     outstat
ED43 20 9AED                jsr     waitecho

ED46 20 75E9                jsr     relall

ED49 A9 00                  lda     #0               ;reset seq # indicators
ED4B 85 42                  sta     cursent
ED4D 85 44                  sta     rnum
ED4F 85 45                  sta     tnum
ED51 85 41                  sta     tcnt
ED53 85 3D                  sta     inform

ED55 20 E3E9                jsr     clrtrbfs         ;make sure the REPT packet is next

ED58 A9 87                  lda     #REPT            ;que the report packet
ED5A 20 46E4                jsr     quepack

ED5D 20 42E2                jsr     wait             ;relinquish control to the transmitter
ED60 4C BFE7                jmp     fmnet            ;on the fmnet queue, so return there

ED63 20 8DED        rdiscon:jsr     waitnet          ;come here if the distant TIE initiated the
                                                     ;discon sequence, and wait until the output
                                                     ;from the network is finished
ED66 A0 A8                  ldy     #mess9           ;inform the user of a remote disconnect
ED68 A9 FE                  lda     #mess9 > 8
ED6A 20 9BE7        3:      jsr     outmess          ;que up the message to be output

ED6D 20 DDED                jsr     outstat          ;que up a stat message also
ED70 20 9AED                jsr     waitecho         ;wait until the echo buffer is empty

ED73 20 75E9                jsr     relall

ED76 A9 00                  lda     #0               ;reset the sequence number indicators
ED78 85 42                  sta     cursent          ;and a couple other things
ED7A 85 44                  sta     rnum
ED7C 85 45                  sta     tnum
ED7E 85 41                  sta     tcnt
ED80 85 3D                  sta     inform

ED82 20 E3E9                jsr     clrtrbfs         ;make sure the REPT packet is next

ED85 A9 87                  lda     #REPT            ;and que up a report datagram
ED87 20 46E4                jsr     quepack

                                                     ;and staying in wait loop will either send it
```

```
ED8A 4C CCEA               jmp     tonet           ;or not send it, but will init the tie in
                                                   ;either case

                   ;**************************************************************************

ED8D A5 2C         waitnet:lda     reccnt          ;get current count of full net receive bufs
ED8F 05 4A                 ora     currcnt         ;get count of chars left in buf
                                                   ;currently being emptied
ED91 F0 06                 beq     2f              ;if both are zero, go return

ED93 20 42E2               jsr     wait            ;else relinquish control to sched
ED96 4C 8DED               jmp     waitnet         ;go check again to see if output from the
                                                   ;network is finished
ED99 60            2:      rts

                   ;**************************************************************************

ED9A A5 15         waitecho:lda    echo.used       ;wait until the echo buffer has been emptied
ED9C F0 06                 beq     2f              ;this indicates that any messages have been
                                                   ;output to the screen
ED9E 20 42E2               jsr     wait            ;relinquish control
EDA1 4C 9AED               jmp     waitecho        ;check again for an empty echo buf

EDA4 60            2:      rts                     ;return when echo buffer is empty

                   ;**************************************************************************

EDA5 A0 9D         conmess:ldy     #mess7          ;come here to output a message indicating
EDA7 A9 FE                 lda     #mess7 > 8      ;that a connection has occurred
EDA9 20 9BE7               jsr     outmess

EDAC A9 01                 lda     #1              ;make sure flow from term
EDAE 85 3A                 sta     fctask          ;is turned on
EDB0 85 04                 sta     outterm

EDB2 60                    rts

                   ;**************************************************************************

                                                   ;remove entry from que for net transmitter
EDB3 C6 1F         decque: dec     quecnt          ;decrement count of things to do
EDB5 A6 29                 ldx     queoutp         ;get current output pointer
EDB7 BC EAE2               ldy     nxtindx[x]      ;use it to get new output pointer
EDBA 84 29                 sty     queoutp         ;save it

EDBC 60                    rts

                   ;**************************************************************************

                                                   ;increment a 3 byte counter
EDBD A9 01         inccount:lda    #1              ;only incrementing counter by 1
EDBF 8D BF02       addcount:sta    temp            ;put increment amount into temp
EDC2 86 5F                 stx    counters        ;store low half of address of counter to
                                                   ;be incremented, high half is set up in init
                                                   ;since all the counters are on same page
EDC4 18                    clc                     ;make sure carry is cleared
```

```

EDC5 A0 00                 ldy     #0              ;only way to do indirect loads is with @[y]
EDC7 B1 5F                 lda     counters@[y]    ;get low byte of counter
EDC9 6D BF02               adc     temp              ;add with carry 1
EDCC 91 5F                 sta     counters@[y]    ;and store it back in low byte of counter

EDCE C8                    iny                     ;increment y to point at middle byte of counter
EDCF B1 5F                 lda     counters@[y]
EDD1 69 00                 adc     #0
EDD3 91 5F                 sta     counters@[y]

EDD5 C8                    iny                     ;increment y to point at high byte of counter
EDD6 B1 5F                 lda     counters@[y]
EDD8 69 00                 adc     #0
EDDA 91 5F                 sta     counters@[y]

EDDC 60                    rts

                   ;**********************************************************************

EDDD A0 CB         outstat:ldy     #messll         ;output the status message to the user
EDDF A9 FE                 lda     #messll > 8
EDE1 20 9BE7               jsr     outmess

EDE4 AD DF02               lda     maxcol          ;maximum collisions on a packet
EDE7 20 B7F4               jsr     outnum2

EDEA A2 08                 ldx     #8
EDEC 8E C002               stx     templ
EDEF A9 20                 lda     #SP
EDF1 20 A7E5       0:      jsr     echoal
EDF4 CE C002               dec     templ
EDF7 D0 F8                 jne     0b

EDF9 AD DE02               lda     totcol + 2      ;total collisions
EDFC 20 B7F4               jsr     outnum2
EDFF AD DD02               lda     totcol + 1
EE02 20 B7F4               jsr     outnum2
EE05 AD DC02               lda     totcol
EE08 20 B7F4               jsr     outnum2

EE0B A0 10                 ldy     #messl2
EE0D A9 FF                 lda     #messl2 > 8
EE0F 20 9BE7               jsr     outmess

EE12 AD DA02               lda     maxlost         ;maximum lost on a packet
EE15 20 B7F4               jsr     outnum2

EE18 A2 08                 ldx     #8
EE1A 8E C002               stx     templ
EE1D A9 20                 lda     #SP
EE1F 20 A7E5       0:      jsr     echoal
EE22 CE C002               dec     templ
EE25 D0 F8                 jne     0b

EE27 AD [illegible]        lda     totlost + 2     ;total lost packets
```

```
EE2A 20 B7F4              jsr     outnum2
EE2D AD D802              lda     totlost + 1
EE30 20 B7F4              jsr     outnum2
EE33 AD D702              lda     totlost
EE36 20 B7F4              jsr     outnum2

EE39 A0 2B                ldy     #mess13
EE3B A9 FF                lda     #mess13 > 8
EE3D 20 9BE7              jsr     outmess

EE40 AD C502              lda     tottrns + 2     ;total successful xmissions
EE43 20 B7F4              jsr     outnum2
EE46 AD C402              lda     tottrns + 1
EE49 20 B7F4              jsr     outnum2
EE4C AD C302              lda     tottrns
EE4F 20 B7F4              jsr     outnum2

EE52 A9 0A                lda     #LF
EE54 4C A7E5              jmp     echoal          ;let "echoal" do the return

```

```
                  ;do all the command interpretation

EE57 A9 04        cmdintrp:lda  #4
EE59 8D 9902               sta  entrysiz         ;set up size of entrys in table

EE5C A9 00                 lda  #0               ;point at beginning of command line
EE5E 8D 9702               sta  cmdpntr

EE61 A9 F6                 lda  #cmdlist         ;get low half of address of command table
EE63 85 5D                 sta  tabpoint         ;store it in low half of pointer
EE65 A9 EE                 lda  #cmdlist > 8     ;get high half of address
EE67 85 5E                 sta  tabpoint + 1     ;store it

EE69 20 85EE               jsr  lookup           ;try to find command

EE6C 29 01                 and  #1
EE6E F0 0E                 beq  notfound         ;command was not found in table

EE70 AD 9802      found:   lda  tabindx          ;come here if the command was found, and get
EE73 0A                    asl                   ;the pointer to the found command, multiply i
EE74 AA                    tax                   ;by 2, and transfer it into the x-reg
EE75 BD B7EE               lda  cmdrout+1[x]     ;use this as an index into a dispatch table
EE78 48                    pha                   ;do the dispatch by pushing the address of
EE79 BD B6EE               lda  cmdrout[x]       ;the found command handling routine onto the
EE7C 48                    pha                   ;stack and then
EE7D 60                    rts                   ;do a rts, which transfers control to that
                                                 ;routine

EE7E A0 2E        notfound:ldy  #mess2           ;come here if command not found, and
EE80 A9 FE                 lda  #mess2 > 8       ;set up a message indicating that to the
EE82 4C 9BE7               jmp  outmess          ;user, outmess will do the return

                  ;**********************************************************************

                                                 ;try to find the command in the list
EE85 A0 00        lookup:  ldy  #0
EE87 8C BF02               sty  temp             ;this is to prime the loop, store it
EE8A 8C 9802               sty  tabindx          ;point at first entry
EE8D 4C A2EE               jmp  1f

EE90 AD BF02      nxtlst:  lda  temp             ;get the current index into the table
EE93 18                    clc                   ;make sure the carry is clear
EE94 6D 9902               adc  entrysiz         ;add size of entry to bump to next command in list
EE97 8D BF02               sta  temp             ;save it
EE9A A8                    tay                   ;put it into the index register
EE9B EE 9802               inc  tabindx          ;point at next entry
EE9E B1 5D                 lda  tabpoint@[y]     ;get the first char in the current command
EEA0 F0 13                 beq  2f               ;if it is a zero, no more commands in table

EEA2 AE 9702      1:       ldx  cmdpntr          ;else set up the y-reg to point into the
EEA5 B1 5D        nxtchar:lda   tabpoint@[y]     ;user command line, then get the next
EEA7 F0 0A                 beq  1f               ;char in the command list, if zero, command
EEA9 DD AE02               cmp  cmdbuf[x]        ;matches, else check char against the
EEAC D0 E2                 bne  nxtlst           ;next user char, if it doesn't match, go
EEAE C8                    iny                   ;set up the next command from the list
EEAF E8                    inx                   ;else bump up both index registers
```

```
EEB0 4C A5EE               jmp     nxtchar         ;and check the next character

EEB3 A9 01         1:      lda     #1
EEB5 60            2:      rts

                   ;************************************************************************

                   ;this is a table of addresses of the routines
                   ;which handle the user commands, it is used in the dispatch

EEB6               cmdrout:addr    craw-1,cnoansi-1,cnolfcr-1,cnoecho-1
EEBE                       addr    cnohup-1,cconn-1,cdiscon-1,crate-1
EEC6                       addr    hdreset-1,cstat-1,cdelay-1,cbreak-1
EECE                       addr    cnobreak-1,cchange-1,cparam-1
                   ;QBF     addr    cqbf-1,crot-1,cnorot-1,cmonitor-1
EED4                       addr    ccooked-1,cansi-1,clfcr-1,cecho-1,chup-1
EEDE                       addr    ctrans-1,cnotrans-1,ctab-1,cnotab-1
EEE6                       addr    cedit-1,cnoedit-1,cmap-1,cnomap-1
EEEE                       addr    cnone-1,ceven-1,codd-1,cany-1
                   ;ENQ     addr    csetdel-1

                   ;this is a table of all the legal commands, they must be zero terminated
                   ;and exactly 4 bytes long including the terminating zero

EEF6               cmdlist:byte    "-li",0         ;char at a time, no line editing
EEFA                       byte    "-an",0         ;no interpretation of chars from net into ANSI 3.64
EEFE                       byte    "-lf",0         ;don't map <lf> -> <cr><lf>
EF02                       byte    "-ec",0         ;turn off local echo
EF06                       byte    "-hu",0         ;turn off auto hangup
EF0A                       byte    "con",0         ;establish connection
EF0E                       byte    "dis",0         ;disconnect
EF12                       byte    "rat",0         ;set uart baud rate
EF16                       byte    "res",0         ;re-initialize the tie
EF1A                       byte    "sta",0         ;display current status
EF1E                       byte    "del",0         ;set up character delay
EF22                       byte    "bre",0         ;set character to break class
EF26                       byte    "-br",0         ;remove character from break class
EF2A                       byte    "cha",0         ;change command character
EF2E                       byte    "par",0         ;print parameter settings
                   ;QBF     byte    "qbf",0         ;send continuous qbf packets to dist tie
                   ;QBF     byte    "rot",0         ;turn on software rotary switch
                   ;QBF     byte    "-ro",0         ;turn off software rotary switch
                   ;QBF     byte    "mon",0         ;switch UB to address-monitor mode
EF32                       byte    "lin",0         ;line at a time, with editing
EF36                       byte    "ans",0         ;interpret characters from net into ANSI 3.64
EF3A                       byte    "lfc",0         ;map <lf> -> <cr><lf>
EF3E                       byte    "ech",0         ;echo locally
EF42                       byte    "hup",0         ;turn on auto hangup
EF46                       byte    "tra",0         ;all characters are transmitted as is
EF4A                       byte    "-tr",0         ;characters are interpreted before xmission
EF4E                       byte    "tab",0         ;output tab character as is
EF52                       byte    "-ta",0         ;map tab character into 8 spaces
EF56                       byte    "edi",0         ;line editing before xmission -- dlc, dll, and dlw
EF5A                       byte    "-ed",0         ;no line editing done
EF5E                       byte    "map",0         ;map lf and ht
EF62                       byte    "-ma",0         ;don't map lf and ht
```

```
EF66                byte    "non",0         ;no parity, don't mask input or output - 8 bit
EF6A                byte    "eve",0         ;even parity
EF6E                byte    "odd",0         ;odd parity
EF72                byte    "any",0         ;accept any parity, output with parity bit 0
            ;ENQ    byte    "set",0         ;set delay after ack was received
EF76                byte    0               ;end list of commands

EF77        spdmsg: byte    LF,"baud rate:  ",0
EF85        spdlist:byte    "110",0         ;set uart to 110 baud
EF89                byte    "150",0         ;set uart to 150 baud
EF8D                byte    "300",0         ;set uart to 300 baud
EF91                byte    "600",0         ;set uart to 600 baud
EF95                byte    "120",0         ;set uart to 1200 baud
EF99                byte    "180",0         ;set uart to 1800 baud
EF9D                byte    "240",0         ;set uart to 2400 baud
EFA1                byte    "360",0         ;set uart to 3600 baud
EFA5                byte    "480",0         ;set uart to 4800 baud
EFA9                byte    "720",0         ;set uart to 7200 baud
EFAD                byte    "960",0         ;set uart to 9600 baud
EFB1                byte    "192",0         ;set uart to 19200 baud
EFB5                byte    0

EFB6        spdconv:byte    $32,$34,$35,$36,$37,$38,$3A,$3B,$3C,$3D,$3E,$3F

EFC2        delchars:byte   "bs",0
EFC5                byte    "ht",0
EFC8                byte    "lf",0
EFCB                byte    "vt",0
EFCE                byte    "ff",0
EFD1                byte    "cr",0
EFD4                byte    0


EFD5        mapchars:byte   BS,HT,LF,VT,FF,CR


EFDB        cmdstr: byte    "rpl",0
EFDF                byte    "dlc",0
EFE3                byte    "dlw",0
EFE7                byte    "dll",0
EFEB                byte    "tnl",0
EFEF                byte    "esc",0
EFF3                byte    "int",0
EFF7                byte    "stp",0
EFFB                byte    "str",0
EFFF                byte    "cmd",0
F003                byte    0

F004        cmdindx:byte    fcrpl,fcdlc,fcdlw,fcdll
F008                byte    fctnl,fcesc,fcint,fcstp
F00C                byte    fcstr,fctie

            ;*********************************************************************

                                            ;change command character
F00E A0 03  cchange:ldy     #3
```

```
3347 F010 20 80F4             jsr     skip            ;skip thru command buffer to set y = pointer
3348 F013 8C 9702             sty     cmdpntr         ;to first char of parameter to be changed
3349
3350 F016 A9 04               lda     #4
3351 F018 8D 9902             sta     entrysiz        ;set table entry size
3352
3353 F01B A9 DB               lda     #cmdstr         ;set up table address
3354 F01D 85 5D               sta     tabpoint
3355 F01F A9 EF               lda     #cmdstr > 8
3356 F021 85 5E               sta     tabpoint + 1
3357
3358 F023 20 85EE             jsr     lookup          ;try to find the first parameter in the table
3359
3360 F026 29 01               and     #1
3361 F028 D0 03               jeq     badparm         ;if not found put out the BAD PARAMETER message
     F02A 4C 42F2
3362
3363 F02D 8A                  txa
3364 F02E A8                  tay
3365 F02F 20 96F4             jsr     skipto          ;skip over 1st parm to SP or HT
3366
3367 F032 C8                  iny                     ;increment pointer twice
3368 F033 C8                  iny
3369 F034 CC AE02             cpy     cmdbuf.inp      ;this should point to end of command
3370 F037 F0 2F               jeq     1f              ;yes
3371
3372 F039 88                  dey
3373 F03A CC AE02             cpy     cmdbuf.inp      ;if want to change nothing
3374 F03D F0 03               jne     badparm         ;if not, bad parameter
     F03F 4C 42F2
3375
3376 F042 AE 9802             ldx     tabindx
3377 F045 BD 04F0             lda     cmdindx[x]
3378 F048 8D C002             sta     templ
3379 F04B C9 14               cmp     #fctie
3380 F04D D0 03               jeq     badparm
     F04F 4C 42F2
3381
3382 F052 A0 00               ldy     #0
3383 F054 B9 7F00        2:   lda     inptab[y]       ;search the key table for the present
3384 F057 29 FE               and     #$FE            ;key with this function
3385 F059 CD C002             cmp     templ
3386 F05C F0 04               jeq     2f
3387
3388 F05E C8                  iny
3389 F05F 10 F3               jpl     2b
3390 F061 60                  rts
3391 F062                2:
3392 F062 A9 00               lda     #0
3393 F064 99 7F00             sta     inptab[y]       ;remove function from table
3394 F067 60                  rts
3395 F068                1:
3396 F068 88                  dey                     ;if OK, then back up one character
3397 F069 B9 AF02             lda     cmdbuf[y]       ;and temp save the new
3398 F06C 8D BF02             sta     temp            ;command character
3399
```

```
F06F AE 9802            ldx     tabindx
F072 BD 04F0            lda     cmdindx[x]
F075 8D C002            sta     templ           ;save which command is being changed

F078 A0 00              ldy     #0
F07A B9 7F00    1:      lda     inptab[y]       ;search the key table for the present
F07D 29 FE              and     #$FE            ;key with this function
F07F CD C002            cmp     templ
F082 F0 06              beq     1f

F084 C8                 iny
F085 10 F3              bpl     1b
F087 4C A7F0            jmp     2f              ;if there isn't one, that's OK too


F08A C0 00      1:      cpy     #0
F08C F0 19              jeq     2f
F08E CC BF02            cpy     temp            ;is the function already mapped to the key
F091 F0 14              beq     2f              ;yes
F093 AE BF02            ldx     temp
F096 B5 7F              lda     inptab[x]       ;has a command assigned to it
F098 29 FE              and     #$FE
F09A F0 03              jne     badparm         ;then that's an error
F09C 4C 42F2

F09F B9 7F00            lda     inptab[y]       ;take the function out for the old
F0A2 29 01              and     #BREAK          ;command character
F0A4 99 7F00            sta     inptab[y]

F0A7 AC BF02    2:      ldy     temp
F0AA B9 7F00            lda     inptab[y]       ;and put it in for the new command char
F0AD 0D C002            ora     templ
F0B0 99 7F00            sta     inptab[y]       ;in the key table

F0B3 AD C002            lda     templ
F0B6 C9 04              cmp     #fcdlc          ;if the del char command was changed
F0B8 D0 06              bne     3f

F0BA AD BF02            lda     temp            ;then change the cmdedit del char too
F0BD 85 7E              sta     cmdedit

F0BF 60                 rts                     ;return

F0C0            3:
F0C0 C9 0A              cmp     #fctnl          ;is it new line command
F0C2 D0 08              jne     3f

F0C4 B9 7F00            lda     inptab[y]
F0C7 09 01              ora     #BREAK          ;make sure it is break class also
F0C9 99 7F00            sta     inptab[y]
F0CC 60         3:      rts

                ;*********************************************************************
                ;ENQcsetdel:
                ;ENQ        ldy     #3
                ;ENQ        jsr     skip            ;skip to first char of first parameter
```

```
                         ;ENQ         sty     cmdpntr         ;store pointer to it
                         ;ENQ         lda     cmdbuf[y]       ;get next char from command buffer
                         ;ENQ         sec
                         ;ENQ         sbc     #'0
                         ;ENQ         jmi     badparm         ;char can't be < ASCII 0
                         ;ENQ         cmp     #10
                         ;ENQ         jcs     badparm         ;or > ASCII 9
                         ;ENQ         sta     ackdelay
                         ;ENQ         rts
                         ;ENQ
                         ;ENQ;*********************************************************************


                                                       ;make a character a "break class" char
 F0CD A0 03              cbreak: ldy     #3
 F0CF 20 96F4                    jsr     skipto        ;skip thru command buffer until a SP or HT

 F0D2 C8                         iny                   ;if we are at the end of the
 F0D3 CC AE02                    cpy     cmdbuf.inp    ;command buffer or beyond
 F0D6 B0 10                      jcs     prntbrk       ;then print current break class characters

 F0D8 BE AF02            0:      ldx     cmdbuf[y]     ;get present function value for this char
 F0DB B5 7F                      lda     inptab[x]     ;in the key table

 F0DD 09 01                      ora     #BREAK        ;make it a break class
 F0DF 95 7F                      sta     inptab[x]     ;and restore it to the table

 F0E1 C8                         iny                   ;bump pointer again
 F0E2 CC AE02                    cpy     cmdbuf.inp    ;if not end of command yet,
 F0E5 90 F1                      bcc     0b            ;get next char to make break class

 F0E7 60                         rts

 F0E8                    prntbrk:
 F0E8 A9 FD                      lda     #brkmsg > 8
 F0EA 20 A7E5                    jsr     echoal
 F0ED A9 70                      lda     #brkmsg
 F0EF 20 A7E5                    jsr     echoal

 F0F2 A2 00                      ldx     #0
 F0F4 A0 00                      ldy     #0
 F0F6 B5 7F              0:      lda     inptab[x]             ;get function out of table
 F0F8 8E BF02                    stx     temp
 F0FB 29 01                      and     #BREAK                ;see if it is break class
 F0FD F0 31                      beq     1f                    ;no

 F0FF AD BF02                    lda     temp                  ;yes, see if it is a control character
 F102 C9 20                      cmp     #$20
 F104 B0 0B                      bcs     2f                    ;no

 F106 A9 5E                      lda     #'©                   ;yes, have it printed readable
 F108 20 A7E5                    jsr     echoal
 F10B AD BF02                    lda     temp
 F10E 18                         clc
 F10F 69 40                      adc     #'@
 F111 20 A7E5            2:      jsr     echoal
```

```
F114 A9 20                 lda     #SP
F116 20 A7E5               jsr     echoal
F119 C8                    iny
F11A C0 28                 cpy     #40             ;echo buffer slots limit of 40 break characters
F11C D0 03                 jeq     3f
F11E 4C DEF2
F121 A5 15                 lda     echo.used               ;how many slots used
F123 C9 4D                 cmp     #77             ;if > 77 and < 80 then insert linefeed for newline
F125 90 09                 bcc     1f
F127 C9 50                 cmp     #80
F129 B0 05                 bcs     1f
F12B A9 0A                 lda     #LF
F12D 20 A7E5               jsr     echoal
F130 AE BF02       1:      ldx     temp
F133 E8                    inx
F134 E0 80                 cpx     #128
F136 D0 BE                 bne     0b

F138 A9 0A                 lda     #LF
F13A 4C A7E5               jmp     echoal
                   ;*************************************************************************


                                                   ;make a char "not a break class"
F13D A0 03         cnobreak:ldy    #3
F13F 20 96F4               jsr     skipto          ;skip thru command buffer until a SP or HT

F142 C8                    iny                     ;bump pointer once
F143 CC AE02               cpy     cmdbuf.inp      ;if at end of buffer or beyond
F146 90 03                 jcs     badparm         ;then bad parameter
F148 4C 42F2

F14B BE AF02       0:      ldx     cmdbuf[y]       ;get present function value for this character
F14E B5 7F                 lda     inptab[x]       ;from key table

F150 29 FE                 and     #$FE            ;and strip the break bit off it
F152 95 7F                 sta     inptab[x]       ;and put it back

F154 C8                    iny
F155 CC AE02               cpy     cmdbuf.inp      ;if any more chars follow in command
F158 90 F1                 bcc     0b              ;go do the same for them

F15A 60                    rts

                   ;********************** ***************************************************

F15B A9 00         chup:   lda     #0              ;turn off the automatic hang-up feature
F15D 85 66                 sta     HUPflg
F15F 60                    rts                     ;and return

                   ;*************************************************************************

F160 A9 01         cnohup: lda     #1              ;turn on the automatic hang-up feature
F162 85 66                 sta     HUPflg
F164 60                    rts                     ;and return

```

```
                   ;**********************************************************************
F165 A2 00         ctrans: ldx     #0              ;set flag indicating all data is to pass untoucjed
F167 86 67                 stx     transflg
F169 60                    rts

                   ;**********************************************************************
F16A A2 01         cnotrans:ldx    #1              ;set flag indicating resume normal mode
F16C 86 67                 stx     transflg
F16E 60                    rts

                   ;**********************************************************************
F16F A2 00         ctab:   ldx     #0              ;set flag outputting tab as tab character
F171 86 68                 stx     tab
F173 60                    rts

                   ;**********************************************************************
F174 A2 01         cnotab: ldx     #1              ;set flag mapping tab into 8 spaces
F176 86 68                 stx     tab
F178 60                    rts

                   ;**********************************************************************
F179 A2 00         cedit:  ldx     #0              ;set flag allowing line editing
F17B 86 69                 stx     edit
F17D 60                    rts

                   ;**********************************************************************
F17E A2 01         cnoedit:ldx     #1              ;don't do line editing
F180 86 69                 stx     edit
F182 60                    rts

                   ;**********************************************************************
F183 A2 00         cmap:   ldx     #0              ;set flag mapping lf and ht
F185 86 6A                 stx     map
F187 60                    rts

                   ;**********************************************************************
F188 A2 01         cnomap: ldx     #1              ;don't map lf and ht
F18A 86 6A                 stx     map
F18C 60                    rts

                   ;**********************************************************************

F18D A0 5E         cstat:  ldy     #mess5          ;output the current connection status to
F18F A9 FE                 lda     #mess5 > 8      ;the screen
F191 20 9BE7               jsr     outmess

F194 A5 59                 lda     distaddh        ;it consists of the local TIE address
F196 20 B7F4               jsr     outnum2
F199 A5 58                 lda     distaddl        ;output actual hexadecimal address
F19B 20 B7F4               jsr     outnum2         ;<nnnn>

F19E A2 0E                 ldx     #14
F1A0 8E C002               stx     templ           ;put out 14 spaces
F1A3 A9 20                 lda     #SP
F1A5               0:
F1A5 20 A7E5               jsr     echoal
```

```
F1A8 CE C002             dec     temp1
F1AB D0 F8               jne     0b

F1AD AD D8FF             lda     SRCaddr         ;it consists of the local TIE address
F1B0 20 B7F4             jsr     outnum2
F1B3 AD D9FF             lda     SRCaddr + 1     ;output actual hexadecimal address
F1B6 20 B7F4             jsr     outnum2         ;<nnnn>

F1B9 A9 0A               lda     #LF
F1BB 20 A7E5             jsr     echoal

F1BE 4C DDED             jmp     outstat         ;output collision and retransmission stats

                ;**********************************************************************

F1C1 A0 03      crate:   ldy     #3              ;skip to first parameter
F1C3 20 80F4             jsr     skip

F1C6 B9 AF02             lda     cmdbuf[y]       ;if end of command buffer
F1C9 F0 77               jeq     badparm         ;then bad parameter

F1CB 8C 9702             sty     cmdpntr         ;else start here to look

F1CE A9 85               lda     #spdlist        ;for a valid speed parameter
F1D0 85 5D               sta     tabpoint
F1D2 A9 EF               lda     #spdlist > 8
F1D4 85 5E               sta     tabpoint + 1

F1D6 20 85EE             jsr     lookup

F1D9 29 01               and     #1              ;if not found in list
F1DB F0 65               jeq     badparm         ;then bad parameter

F1DD AE 9802             ldx     tabindx         ;if good, then use table index
F1E0 BD B6EF             lda     spdconv[x]      ;to set the new speed
F1E3 AE 0310             ldx     a10cntr
F1E6 AE 0210             ldx     a10mode
F1E9 8D 0210             sta     a10mode         ;in the uart
F1EC 60                  rts

                ;**********************************************************************

F1ED A9 00      cecho:   lda     #0              ;turn local echoing on
F1EF 85 65               sta     echo.off
F1F1 60                  rts                     ;and return

                ;**********************************************************************

F1F2 A9 01      cnoecho:lda      #1              ;turn local echoing off
F1F4 85 65               sta     echo.off
F1F6 60                  rts                     ;and return

                ;**********************************************************************

F1F7 A9 01      craw:    lda     #1              ;switch to character at a time mode
F1F9 A2 00               ldx     #0              ;always interpret strings from echo buf
```

```
F1FB 4C 03F2               jmp     2f              ;save some space

F1FE A9 00         ccooked:lda     #0              ;switch to line at a time mode
F200 AA                    tax                     ;always interpret strings
F201 85 65         1:      sta     echo.off        ;alsways echo when in line mode
F203 85 62         2:      sta     rawcook
F205 85 7B                 sta     instat1
F207 86 7D                 stx     outstat2        ;always map during echoed(editing)strings
F209 60                    rts                     ;and return

                   ;***********************************************************************

F20A A2 00         cansi:  ldx     #0              ;interpret and map data from network
F20C 4C 11F2               jmp     0f              ;save some space

                   ;***********************************************************************
F20F               cnoansi:
F20F A2 04                 ldx     #4              ;turn off mapping
F211 86 63         0:      stx     outstat1
F213 86 7A                 stx     savstat
F215 60                    rts

                   ;***********************************************************************

F216 A9 00         clfcr:  lda     #0              ;map <lf> -> <cr><lf>
F218 4C 1DF2               jmp     0f

                   ;***********************************************************************
F21B               cnolfcr:
F21B A9 01                 lda     #1              ;turn off map <lf> -> <cr><lf>
F21D 85 64         0:      sta     lfcr
F21F 60                    rts

                   ;***********************************************************************

F220 A2 4E         cany:   ldx     #$4E            ;set to 8 bit, no parity, 1 stop bit
F222 A0 03                 ldy     #ANY
F224 4C 39F2               jmp     setpar

F227 A2 4E         cnone:  ldx     #$4E            ;set to 8 bit, no parity, 1 stop bit
F229 A0 00                 ldy     #NONE
F22B 4C 39F2               jmp     setpar

F22E A2 7A         ceven:  ldx     #$7A            ;set to 7 bit, even parity, 1 stop bit
F230 A0 01                 ldy     #EVEN
F232 4C 39F2               jmp     setpar

F235 A2 5A         codd:   ldx     #$5A            ;set to 7 bit, odd parity, 1 stop bit
F237 A0 02                 ldy     #ODD

F239 AD 0310       setpar: lda     a10cntr         ;reset 2651
F23C 8E 0210               stx     a10mode
F23F 84 79                 sty     partype         ;set parity paramater, mask both inp and outp
F241 60                    rts

                   ;***********************************************************************
```

```

F242 A0 BC        badparm:ldy   #mess10         ;send BAD PARAMETER to terminal
F244 A9 FE                lda   #mess10 > 8
F246 4C 9BE7              jmp   outmess

                  ;*************************************************************************

F249              cparam:
F249 A9 00                lda   #0
F24B 48                   pha                           ;save index on stack
F24C A2 00                ldx   #0
F24E 8E BF02      1:      stx   temp
F251 B5 62                lda   termparams[x]           ;show - if necessary
F253 F0 05                beq   2f
F255 A9 2D                lda   #'-
F257 20 A7E5              jsr   echoal
F25A 68           2:      pla                   ;get current index into parameter messages
F25B 48                   pha                   ;keep saving it
F25C 20 02F3              jsr   cmdmess                 ;print corresponding command
F25F 68                   pla
F260 18                   clc
F261 69 06                adc   #6                      ;point to next command string
F263 48                   pha                           ;save it
F264 AE BF02              ldx   temp
F267 E8                   inx
F268 E0 09                cpx   #PARM.MAX
F26A D0 E2                bne   1b
F26C 68                   pla
F26D A4 79                ldy   partype                 ;print what kind of parity
F26F F0 06                jeq   1f
F271 18           0:      clc
F272 69 05                adc   #5
F274 88                   dey
F275 D0 FA                bne   0b

F277 20 02F3      1:      jsr   cmdmess
F27A A9 FD                lda   #par.str > 8
F27C 20 A7E5              jsr   echoal
F27F A9 34                lda   #par.str
F281 20 A7E5              jsr   echoal

F284 A9 EF                lda   #spdmsg > 8
F286 20 A7E5              jsr   echoal
F289 A9 77                lda   #spdmsg
F28B 20 A7E5              jsr   echoal

F28E AE 0310              ldx   ai0cntr
F291 AD 0210              lda   ai0mode
F294 AD 0210              lda   ai0mode
F297 29 0F                and   #$0F            ;speed pattern is lower byte of MODE 2 reg.
F299 AA                   tax
F29A A9 88                lda   #outspd
F29C CA           2:      dex
F29D 30 [illegible]       bmi   1f
F29F 18                   clc
F2A0 69 06                adc   #6
```

```
F2A2 4C 9CF2            jmp     2b
F2A5 A8         1:      tay
F2A6 A9 00              lda     #0
F2A8 69 FC              adc     #outspd > 8
F2AA 20 A7E5            jsr     echoal
F2AD 98                 tya
F2AE 20 A7E5            jsr     echoal
F2B1 A5 67              lda     transflg          ;if in transparent mode
F2B3 F0 48              beq     0f                ;don't print key functions
F2B5 A9 FD              lda     #funcmsg > 8      ;print function characters
F2B7 20 A7E5            jsr     echoal
F2BA A9 89              lda     #funcmsg
F2BC 20 A7E5            jsr     echoal

F2BF A0 14              ldy     #fctie            ;print characters for func mnemonics
F2C1 8C C002    1:      sty     templ
F2C4 A0 7F              ldy     #127
F2C6 B9 7F00    2:      lda     inptab[y]         ;find corresponding character in inptab
F2C9 29 FE              and     #$FE
F2CB CD C002            cmp     templ
F2CE F0 0E              beq     3f                ;found it

F2D0 88                 dey
F2D1 10 F3              bpl     2b                ;if haven't searched all, keep trying
F2D3 A9 20              lda     #SP
F2D5 20 A7E5            jsr     echoal
F2D8 20 A7E5            jsr     echoal            ;function not defined, print spaces
F2DB 4C E2F2            jmp     4f

F2DE 98         3:      tya
F2DF 20 2CE2            jsr     prntchr

F2E2 A9 20      4:      lda     #SP
F2E4 20 A7E5            jsr     echoal
F2E7 20 A7E5            jsr     echoal

F2EA AC C002            ldy     templ             ;restore function value
F2ED 88                 dey
F2EE 88                 dey
F2EF F0 0C              beq     0f                ;thats all of them

F2F1 C0 08              cpy     #fcrpl            ;if in -edit, don't print for rpl, dll,
                                                  ;dlw, and dlc
F2F3 D0 CC              jne     1b
F2F5 A5 69              lda     edit
F2F7 D0 04              jne     0f

F2F9 A5 62              lda     rawcook           ;if in -line, no editting features work
F2FB F0 C4              jeq     1b

F2FD            0:
F2FD A9 0A              lda     #LF
F2FF 4C A7E5            jmp     echoal



```

```
F302 18                cmdmess: clc                 ; print message selected from cmdlist
F303 69 EA                      adc     #parms       ; pointer comes in AC
F305 A8                         tay
F306 A9 00                      lda     #0
F308 69 FC                      adc     #parms > 8
F30A 20 A7E5                    jsr     echoal
F30D 98                         tya
F30E 20 A7E5                    jsr     echoal
F311 60                         rts

F312 18                strmess: clc                 ; print selected msg from cmdstr
F313 69 DB                      adc     #cmdstr      ; pointer comes in AC
F315 A8                         tay
F316 A9 00                      lda     #0
F318 69 EF                      adc     #cmdstr > 8
F31A 20 A7E5                    jsr     echoal
F31D 98                         tya
F31E 20 A7E5                    jsr     echoal
F321 60                         rts


                       ;*********************************************************************

                                                     ;change # of delays for a char
F322 A0 03             cdelay: ldy      #3
F324 20 80F4                    jsr     skip         ;skip to first char of first parameter
F327 8C 9702                    sty     cmdpntr      ;store pointer to it

F32A A9 03                      lda     #3
F32C 8D 9902                    sta     entrysiz     ;set table entry size for lookup

F32F A9 C2                      lda     #delchars    ;set up address of table of delay chars
F331 85 5D                      sta     tabpoint
F333 A9 EF                      lda     #delchars > 8
F335 85 5E                      sta     tabpoint + 1

F337 20 85EE                    jsr     lookup       ;and try to find parameter in the table

F33A 29 01                      and     #1
F33C F0 6C                      jeq     prntdel

F33E 8A                         txa
F33F A8                         tay
F340 20 80F4                    jsr     skip         ;skip to first char of 2nd parameter

F343 A2 02                      ldx     #2           ;limit length of 2nd parm to 2 digits
F345 A9 00                      lda     #0           ;zero temp area for # of delays
F347 8D BF02                    sta     temp
F34A 4C 6AF3                    jmp     2f

F34D 0A                1:       asl                  ;multiply last digit by 10
F34E 90 03                      jcs     badparm      ;on overflow, bad parameter
F350 4C 42F2
F353 0A                         asl
F354 90 03                      jcs     badparm
F356 4C 42F2
```

```
3899 F359 6D BF02               adc     temp
3900 F35C 90 03                 jcs     badparm
     F35E 4C 42F2
3901 F361 0A                    asl
3902 F362 90 03                 jcs     badparm
     F364 4C 42F2
3903 F367 8D BF02               sta     temp            ;and restore it to running delay count
3904
3905 F36A B9 AF02       2:      lda     cmdbuf[y]       ;get next char from command buffer
3906 F36D F0 2E                 beq     3f              ;if zero, then end of buffer
3907 F36F 38                    sec
3908 F370 E9 30                 sbc     #'0
3909 F372 10 03                 jmi     badparm         ;char can't be < ASCII 0
     F374 4C 42F2
3910 F377 C9 0A                 cmp     #10
3911 F379 90 03                 jcs     badparm         ;or > ASCII 9
     F37B 4C 42F2
3912
3913 F37E 6D BF02               adc     temp            ;add this digit into running # of delays
3914 F381 90 03                 jcs     badparm         ;on overflow, bad parameter
     F383 4C 42F2
3915 F386 8D BF02               sta     temp
3916
3917 F389 C8                    iny
3918 F38A B9 AF02               lda     cmdbuf[y]       ;get next char in command buffer
3919 F38D F0 0E                 beq     3f              ;if zero, then end of buffer
3920
3921 F38F AD BF02               lda     temp            ;load up running # of delays
3922 F392 CA                    dex                     ;only allow 2 digits in 2nd parmeter
3923 F393 D0 B8                 bne     1b
3924
3925 F395 B9 AF02               lda     cmdbuf[y]       ;if more than two,
3926 F398 F0 03                 jne     badparm         ;then bad parameter
     F39A 4C 42F2
3927
3928 F39D AE 9802       3:      ldx     tabindx
3929 F3A0 BC D5EF               ldy     mapchars[x]     ;get index to char we changed delays on
3930 F3A3 AD BF02               lda     temp            ;get new # of delays
3931 F3A6 99 6B00               sta     delays[y]       ;store it in the table
3932
3933 F3A9 60                    rts                     ;and return
3934
3935 F3AA               prntdel:
3936 F3AA A9 FD                 lda     #delmsg > 8             ;print message for delay padding
3937 F3AC 20 A7E5               jsr     echoal
3938 F3AF A9 3D                 lda     #delmsg
3939 F3B1 20 A7E5               jsr     echoal
3940
3941 F3B4 A2 00                 ldx     #0
3942 F3B6 8E C102       1:      stx     temp4
3943 F3B9 BC D5EF               ldy     mapchars[x]
3944 F3BC B9 6B00               lda     delays[y]
3945 F3BF 20 B7F4               jsr     outnum2
3946 F3C2 A9 20                 lda     #SP
3947 F3C4 20 A7E5               jsr     echoal
3948 F3C7 EE C102               inc     temp4
```

```
3949 F3CA AE C102               ldx     temp4
3950 F3CD E0 06                 cpx     #6
3951 F3CF D0 E5                 bne     lb
3952 F3D1 A9 0A                 lda     #LF
3953 F3D3 20 A7E5               jsr     echoal
3954 F3D6 60                    rts
3955
3956                    ;***********************************************************************
3957
3958 F3D7 AD 9602       cdiscon:lda     netstate        ;make sure we're connected
3959 F3DA C9 18                 cmp     #CONNECT
3960 F3DC D0 0A                 bne     lf              ;before sending a DISCON packet
3961
3962 F3DE A9 01                 lda     #1              ;set flag to
3963 F3E0 85 3C                 sta     sendisc         ;send the DISC packet
3964 F3E2 85 06                 sta     netwrite        ;and to activate the net write process
3965
3966 F3E4 A9 A0                 lda     #160            ;set DISC timer to wait upto " 10 sec
3967 F3E6 85 51                 sta     distimer        ;for the data que to empty
3968
3969 F3E8 60            l:      rts
3970
3971                    ;***********************************************************************
3972
3973                                                    ;come here to handle connect request from user
3974 F3E9 AD 9002       cconn:  lda     CONNstate       ;see if connection sequence is in progress
3975 F3EC F0 07                 beq     lf              ;if not, continue
3976
3977                    ;QBF    lda     #0              ;can't monitor in a connected state
3978                    ;QBF    sta     monaddr         ;reset monitor flag
3979                    ;QBF
3980
3981 F3EE A0 F5                 ldy     #alrcon         ;else inform the user he cannot initiate
3982 F3F0 A9 FD                 lda     #alrcon > 8     ;a connection at this time
3983 F3F2 4C 9BE7               jmp     outmess         ;outmess does the return
3984
3985 F3F5 A0 03         l:      ldy     #3
3986 F3F7 20 80F4               jsr     skip            ;skip to address
3987
3988 F3FA A2 04                 ldx     #4              ;the address must be 4 hexidecimal digits long
3989 F3FC 38            4:      sec                     ;make sure carry is set
3990 F3FD E9 30                 sbc     #'0             ;if the digit is < 0
3991 F3FF 30 78                 jmi     badaddr         ;go inform user
3992
3993 F401 C9 0A                 cmp     #10             ;else if the digit is <= 9
3994 F403 90 0B                 bcc     lf              ;ok, go set up for next one
3995
3996 F405 C9 31                 cmp     #$31            ;else see if the digit is >= a
3997 F407 90 70                 jcc     badaddr         ;if not, go inform user
3998
3999 F409 C9 37                 cmp     #$37            ;else see if the digit is <= f
4000 F40B B0 6C                 jcs     badaddr         ;if not, go inform user
4001
4002 F40D 38                    sec                     ;else, set carry
4003 F40E E9 27                 sbc     #$27            ;and subtract #$27, to make it real number
4004
```

```
F410 99 AF02       1:      sta     cmdbuf[y]       ;store real number in the same position
F413 C8                    iny                     ;bump pointer to next digit
F414 B9 AF02               lda     cmdbuf[y]       ;get it
F417 CA                    dex                     ;decrement counter
F418 DO E2                 bne     4b              ;if non-zero, go handle digit

F41A C9 00                 cmp     #0              ;else if we have 4 digits, make sure no more
F41C DO 5B                 bne     badaddr         ;on command line; if so, inform user of
                                                   ;bad address
                   ;QBF     lda     monaddr         ;address has checked out OK
                   ;QBF     bne     setmon          ;if monitor mode go set the new address
                   ;QBF

F41E 20 A4F4               jsr     buildbyt        ;else, gather up 2 digits and build a byte
F421 85 58                 sta     distaddl        ;store it as low half of address

F423 20 A4F4               jsr     buildbyt        ;get the other 2 digits
F426 85 59                 sta     distaddh        ;and store as the high half of the address

F428 CD D8FF               cmp     SRCaddr         ;compare it with address of local TIE
F42B DO 14                 bne     1f              ;if they are the same, inform user
F42D A5 58                 lda     distaddl        ;that he can't establish a connection
F42F CD D9FF               cmp     SRCaddr + 1     ;to himself
F432 DO OD                 bne     1f              ;else go finish up

F434 A9 00                 lda     #0              ;set address field back to zero
F436 85 58                 sta     distaddl
F438 85 59                 sta     distaddh

F43A A0 42                 ldy     #mess4          ;output the message to the user
F43C A9 FE                 lda     #mess4 > 8
F43E 4C 9BE7               jmp     outmess         ;outmess does the return

F441               1:
F441               ;QBF     lda     norotary        ;should rotary be set up?
F441               ;QBF     bne     3f              ;no, skip
F441               ;QBF
F441 A5 58                 lda     distaddl        ;see if the address if that of a host
F443 29 FO                 and     #$FO            ;<nn0x> or <nn8x>
F445 FO 04                 beq     2f

F447 C9 80                 cmp     #$80
F449 DO OE                 bne     3f

F44B A8            2:      tay                     ;come here to set up the rotary for a host
F44C A5 58                 lda     distaddl        ;use the low order 4 bits as an index into
F44E 29 OF                 and     #$OF            ;a table
F450 AA                    tax
F451 BD B4EA               lda     hostsiz[x]      ;this table contains the number of TIEs on
F454 8D 9A02               sta     hostconn        ;the host
F457 84 58                 sty     distaddl        ;store the low half of the address <nnn0>

F459 A5 59         3:      lda     distaddh        ;get the high half of the address
F45B 8D 0102               sta     trbuf0 + dstaddr        ;store it in each of the transmit
F45E 8D 0103               sta     trbuf1 + dstaddr        ;buffers
F461 8D 0104               sta     trbuf2 + dstaddr
```

```

F464 A5 58               lda     distaddl        ;get the low half of the address
F466 8D 0202             sta     trbuf0 + dstaddr + 1    ;store it in each of the transmit
F469 8D 0203             sta     trbuf1 + dstaddr + 1    ;buffers
F46C 8D 0204             sta     trbuf2 + dstaddr + 1

F46F A9 01               lda     #BUSY           ;set state to connection in progress
F471 8D 9002             sta     CONNstate

F474 A9 04               lda     #CONN           ;send a CONN packet to the address indicated
F476 4C 46E4             jmp     quepack         ;quepack does the return

                ;QBFsetmon:      jsr     buildbyt
                ;QBF     eor     #$FF
                ;QBF     sta     TIEaddrl        ;set the new address for UB to recognize
                ;QBF
                ;QBF     jsr     buildbyt
                ;QBF     eor     #$FF
                ;QBF     sta     TIEaddrh
                ;QBF
                ;QBF     rts
                ;QBF

                ;**********************************************************************

F479            badaddr:
F479            ;QBF     lda     #0
F479            ;QBF     sta     monaddr         ;address didn't check, leave poss monitor mode
F479            ;QBF
F479 A0 21               ldy     #mess1          ;indicate to user that the address he specified
F47B A9 FE               lda     #mess1 > 8      ;is illegal
F47D 4C 9BE7             jmp     outmess         ;outmess does the return

                ;**********************************************************************

F480 20 96F4    skip:    jsr     skipto          ;skip thru command buffer until a SP, HT or $00

F483 B9 AF02             lda     cmdbuf[y]       ;if $00, then just return
F486 F0 0C               beq     4f

F488 C8         3:       iny                     ;now scan across the user command line
F489 B9 AF02             lda     cmdbuf[y]       ;until anything but a space or
F48C C9 20               cmp     #SP             ;horizontal tab is found
F48E F0 F8               beq     3b
F490 C9 09               cmp     #HT
F492 F0 F4               beq     3b

F494 60         4:       rts

                ;**********************************************************************

F495 C8         1:       iny                     ;the user command line
F496 B9 AF02    skipto:  lda     cmdbuf[y]       ;until a space or a horizontal tab is
[illegible]              beq     2f
[illegible]              cmp     #SP             ;found
[illegible]              beq     2f
```

```
F49F C9 09              cmp     #HT
F4A1 D0 F2              bne     1b

F4A3 60         2:      rts

                ;***********************************************************************

F4A4 88         buildbyt:dey                    ;takes two hexadecimal digits and
F4A5 B9 AF02            lda     cmdbuf[y]       ;builds a byte from them
F4A8 8D BF02            sta     temp            ;go from the least significant to the most
F4AB 88                 dey                     ;significant, first store least signicant
F4AC B9 AF02            lda     cmdbuf[y]       ;away, then get the most significant
F4AF 0A                 asl
F4B0 0A                 asl                     ;shift if to the left 4 times, to get
F4B1 0A                 asl                     ;it into the high half of the byte
F4B2 0A                 asl
F4B3 0D BF02            ora     temp            ;now or the least signifcant digit into
F4B6 60                 rts                     ;the low half of the byte, and return

                ;***********************************************************************

F4B7 8D BF02    outnum2:sta     temp            ;output a byte as 2 hexadecimal digits
                                                ;first save it
F4BA 4A                 lsr                     ;then shift it to the right 4 times
F4BB 4A                 lsr                     ;this moves the high half into the low half
F4BC 4A                 lsr
F4BD 4A                 lsr
F4BE 20 C6F4            jsr     makASCI         ;go convert it to ASCII, and output it

F4C1 AD BF02            lda     temp            ;retreive the byte
F4C4 29 0F              and     #$0F            ;get rid of all but low half of byte


F4C6 C9 0A      makASCI:cmp     #10             ;if digit is < 10
F4C8 90 03              bcc     1f              ;continue

F4CA 18                 clc                     ;else clear carry and
F4CB 69 07              adc     #$07            ;add 7 since it is in the range <A - F>

F4CD 18         1:      clc                     ;clear the carry
F4CE 69 30              adc     #$30            ;and make it an ASCII digit

F4D0 20 A7E5            jsr     echoal          ;stick it in echo buffer, to be output
F4D3 60                 rts                     ;and return

                ;***********************************************************************
```

```
F800          org     $F800

              ;This table contains the definitions of the distance into the dispatch
              ; routine jump table, "termtab".

0000          ffnop = 0                ;ignore char
0004          ffctl = 4                ;output char, don't bump position
0006          ffesc = 6                ;escape, seq to follow
0008          ffcsi = %10              ;escape followed by a '['
000A          ffnl   = %12             ;new line
000C          ffcr   = %14             ;carriage return
000E          ffht   = %16             ;horz tab
0010          ffbs   = %20             ;backspace
0012          ffff   = %22             ;new page
0014          ffvt   = %24             ;vert tab
0016          ffind = %26              ;index
0018          ffri   = %30             ;reverse index
001A          ffpld = %32              ;partial line down
001C          ffplu = %34              ;partial line up
001E          ffhpa = %36              ;horz position absolute
0020          ffhpr = %40              ;horz position relative
0022          ffvpa = %42              ;vert position absolute
0024          ffvpr = %44              ;vert position relative
0026          ffhvp = %46              ;horz and vert position (absolute)
0028          ffsgr = %50              ;select graphic rendition
002A          ffech = %52              ;erase characters
002C          ffel   = %54             ;erase in line
002E          ffed   = %56             ;erase in display
0030          ffdch = %60              ;delete characters
0032          ffdl   = %62             ;delete lines
0034          ffich = %64              ;insert (erased) characters
0036          ffil   = %66             ;insert (erased) lines
0038          ffcuu = %70              ;cursor up
003A          ffcud = %72              ;cursor down
003C          ffcuf = %74              ;cursor forward (right)
003E          ffcub = %76              ;cursor backward (left)
0040          ffcnl = %100             ;cursor next line
0042          ffcpl = %102             ;cursor previous line
0044          ffcha = %104             ;cursor horz absolute
0046          ffcup = %106             ;cursor position (x and y)
0048          ffsu   = %110            ;scroll up
004A          ffsd   = %112            ;scroll down
004C          ffnp   = %114            ;next page
004E          ffpp   = %116            ;previous page
0050          ffhts = %120             ;horizontal tab set
0052          fftbc = %122             ;clear tab stops
0054          ffcht = %124             ;cursor horizontal tabulation
0056          ffcbt = %126             ;cursor backward tabulation

              ;control character to function mapping table
F800          cntltab:
F800                  byte    ffctl,ffctl,ffctl,ffctl ; <nul>,<soh>,<stx>,<etx>
F804                  byte    ffctl,ffctl,ffctl,ffctl ; <eot>,<enq>,<ack>,<bel>
F808                  byte    ffbs,ffht,ffctl,ffctl    ; <bs> ,<ht> ,<lf> ,<vt>
F80C                  byte    ffctl,ffcr,ffctl,ffctl  ; <ff> ,<cr> ,<so> ,<si>
F810                  byte    ffctl,ffctl,ffctl,ffctl ; <dle>,<dc1>,<dc2>,<dc3>
```

```
F814                 byte    ffctl,ffctl,ffctl,ffctl ; <dc4>,<nak>,<syn>,<etb>
F818                 byte    ffctl,ffctl,ffctl,ffesc ; <can>,<em> ,<sub>,<esc>
F81C                 byte    ffctl,ffctl,ffctl,ffctl ; <fs> ,<gs> ,<rs>, <us>

             ;additional controls mapped from <esc> <column 3,4,5>
             ;        the <column 3> controls are private escapes
             ;        the <column 4,5> are Cl controls
F820         cltab:  byte    ffnop,ffnop,ffnop,ffnop         ; '@','A','B','C'
F824                 byte    ffind,ffnl,ffnop,ffnop          ; 'D','E','F','G'
F828                 byte    ffhts,ffnop,ffnop,ffpld         ; 'H','I','J','K'
F82C                 byte    ffplu,ffri,ffnop,ffnop          ; 'L','M','N','O'
F830                 byte    ffnop,ffnop,ffnop,ffnop         ; 'P','Q','R','S'
F834                 byte    ffnop,ffnop,ffnop,ffnop         ; 'T','U','V','W'
F838                 byte    ffnop,ffnop,ffnop,ffcsi         ; 'X','Y','Z','['
F83C                 byte    ffnop,ffnop,ffnop,ffnop         ; '®',']','©','_'

             ; additional controls introduced by <esc> <csi> p... f
             ;       where the p are in <column 3> and
             ;        the f are in <columns 4,5,6,7>
F840         csitab:
F840                 byte    ffich,ffcuu,ffcud,ffcuf         ; '@','A','B','C'
F844                 byte    ffcub,ffcnl,ffcpl,ffcha         ; 'D','E','F','G'
F848                 byte    ffcup,ffcht,ffed,ffel           ; 'H','I','J','K'
F84C                 byte    ffil,ffdl,ffnop,ffnop           ; 'L','M','N','O'
F850                 byte    ffdch,ffnop,ffnop,ffsu          ; 'P','Q','R','S'
F854                 byte    ffsd,ffnp,ffpp,ffnop            ; 'T','U','V','W'
F858                 byte    ffech,ffnop,ffcbt,ffnop         ; 'X','Y','Z','['
F85C                 byte    ffnop,ffnop,ffnop,ffnop         ; '®',']','©','_'
F860                 byte    ffhpa,ffhpr,ffnop,ffnop         ; '°','a','b','c'
F864                 byte    ffvpa,ffvpr,ffhvp,fftbc         ; 'd','e','f','g'
F868                 byte    ffnop,ffnop,ffnop,ffnop         ; 'h','i','j','k'
F86C                 byte    ffnop,ffsgr,ffnop,ffnop         ; 'l','m','n','o'
```

```
        F8A0        org             $F8A0
                ;routines to interpret output to CRT and printing terminal

F8A0            prompt: byte    CR,LF,"(Terminal TIE, 840627) ADDR ",0

        F900            org             $F900

                        nbin

                ;this table contains the addresses of all the output function routines
                ;it is used in the dispatch routine

F900            termtab:addr    xnop-1,xshoit-1,xctl-1,xesc-1   ;ffnop,sshoit,ffctl,ffesc
F908                    addr    xcsi-1,xnl-1,xcr-1,xht-1        ;ffcsi,ffnl,ffcr,ffht
F910                    addr    xbs-1,xnop-1,xnop-1,xnop-1      ;ffbs,ffff,ffvt,ffind
F918                    addr    xnop-1,xnop-1,xnop-1,xnop-1     ;ffri,ffpld,ffplu,ffhpa
F920                    addr    xnop-1,xnop-1,xnop-1,xnop-1     ;ffhpr,ffvpa,ffvpr,ffhvp
F928                    addr    xnop-1,xech-1,xnop-1,xnop-1     ;ffsgr,ffech,ffel,ffed
F930                    addr    xnop-1,xnop-1,xnop-1,xnop-1     ;ffdch,ffdl,ffich,ffil
F938                    addr    xnop-1,xcud-1,xnop-1,xcub-1     ;ffcuu,ffcud,ffcuf,ffcub
F940                    addr    xcnl-1,xnop-1,xnop-1,xnop-1     ;ffcnl,ffcpl,ffcha,ffcup
F948                    addr    xnop-1,xnop-1,xnop-1,xnop-1     ;ffsu,ffsd,ffnp,ffpp
F950                    addr    xnop-1,xnop-1,xcht-1,xnop-1     ;ffhts,fftbc,ffcht,ffcbt

                ;these routines output chars, or cursor positioning information to the terminal
                ;x-reg contains the character to be handled
                ;ieol is incremented or decremented appropriately
                ;state 1 is entered if char is an <esc>

F958 20 0AE7    xshoit: jsr     putout          ;output control char as is

F95B A9 00      xnop:   lda     #0              ;don't output anything
F95D 60                 rts


F95E 20 0AE7    xctl:   jsr     putout          ;output control char as is

F961 A9 00              lda     #0
F963 60                 rts


F964 A9 01      xesc:   lda     #1              ;<esc>, set to state 1
F966 60                 rts


F967 A9 00      xcsi:   lda     #0              ;in state 1, set to state 2, and zero parms
F969 85 16              sta     parm1
F96B 85 17              sta     parm2
F96D 85 18              sta     parm3
F96F 85 19              sta     parm4
F971 85 1A              sta     parm5
F973 85 1B              sta     parm6
F975 85 1C              sta     parm7
F977 85 1D              sta     parm8
F979 8D 0202            sta     parmindx

```

```
F97C A9 02                lda     #2
F97E 60                   rts


F97F A2 0D       xnl:     ldx     #CR             ;<lf> -> <cr>,<lf>
F981 20 98F9              jsr     zeropntr
F984 20 0AE7              jsr     putout

F987 A2 0A                ldx     #LF
F989 20 0AE7              jsr     putout          ;output <cr>, zero lcol,ocol

F98C A9 00                lda     #0
F98E 60                   rts


F98F 20 98F9     xcr:     jsr     zeropntr
F992 20 0AE7              jsr     putout

F995 A9 00                lda     #0
F997 60                   rts


F998 A9 00       zeropntr:lda     #0
F99A 85 1E                sta     lcol
F99C 8D A002              sta     tabx

F99F 60                   rts


F9A0 C6 1E       xbs:     dec     lcol
F9A2 20 0AE7              jsr     putout          ;output a <bs>

F9A5 A9 00                lda     #0
F9A7 60                   rts


F9A8 A5 1E       xht:     lda     lcol
F9AA 8D C002              sta     templ

F9AD 18                   clc
F9AE 69 08                adc     #8              ;compute number of cursor positions generated
F9B0 29 F8                and     #%0370
F9B2 85 1E                sta     lcol

F9B4 38                   sec
F9B5 ED C002              sbc     templ

F9B8 AC A002              ldy     tabx
F9BB 99 A102              sta     tabs[y]         ;store in next slot in tab table
F9BE EE A002              inc     tabx

F9C1 20 0AE7              jsr     putout

F9C4 A9 00                lda     #0
F9C6 60                   rts
```

```

F9C7 A2 20      xech:   ldx     #SP             ;erase n chars, parml contains n
F9C9 A4 16              ldy     parml
F9CB 20 8CE7            jsr     adjy01          ;both 0 and 1 mean 1
F9CE 20 86E7            jsr     reptc0

F9D1 A2 08              ldx     #BS
F9D3 A4 16              ldy     parml
F9D5 20 8CE7            jsr     adjy01
F9D8 20 86E7            jsr     reptc0
F9DB A9 00              lda     #0
F9DD 60                 rts

F9DE A2 08      xcub:   ldx     #BS             ;back up cursor n chars, parml contains n
F9E0 A4 16              ldy     parml
F9E2 20 8CE7            jsr     adjy01          ;both 0 and 1 = 1 step
F9E5 20 0AE7    2:      jsr     putout
F9E8 C6 1E              dec     lcol
F9EA 88                 dey
F9EB D0 F8              bne     2b

F9ED A9 00              lda     #0
F9EF 60                 rts


F9F0 A2 0D      xcnl:   ldx     #CR             ;move cursor next line, do a <cr> first
F9F2 20 98F9            jsr     zeropntr
F9F5 20 0AE7            jsr     putout

F9F8 A2 0A      xcud:   ldx     #LF             ;move cursor down n lines, parml contains n
F9FA A4 16              ldy     parml
F9FC 20 80E7            jsr     reptcl

F9FF A9 00              lda     #0
FA01 60                 rts


FA02            xcht:
FA02 A5 16              lda     parml           ;cursor horizontal tabulation
FA04 D0 02              jne     1f

FA06 E6 16              inc     parml
FA08 20 A8F9    1:      jsr     xht             ;go to at least next tab stop
FA0B C6 16              dec     parml
FA0D D0 F9              jne     1b

FA0F A9 00              lda     #0
FA11 60                 rts

                ;*************************************************************************
          0050  LINSIZ = 80                     ;size of screen line


```

```

      FC80           org     $FC80

FC80           spdtab: byte    $32,$35,$36,$37,$38,$3A,$3C,$3E ;table of speeds to search
FC88           outspd:
FC88                   byte    "50  ",LF,0
FC8E                   byte    "75  ",LF,0
FC94                   byte    "110 ",LF,0            ;output parameter as 110 baud
FC9A                   byte    "134 ",LF,0
FCA0                   byte    "150 ",LF,0            ;output parameter as 150 baud
FCA6                   byte    "300 ",LF,0            ;output parameter as 300 baud
FCAC                   byte    "600 ",LF,0            ;output parameter as 600 baud
FCB2                   byte    "1200",LF,0            ;output parameter as 1200 baud
FCB8                   byte    "1800",LF,0            ;output parameter as 1800 baud
FCBE                   byte    "2000",LF,0
FCC4                   byte    "2400",LF,0            ;output parameter as 2400 baud
FCCA                   byte    "3600",LF,0            ;output parameter as 3600 baud
FCD0                   byte    "4800",LF,0            ;output parameter as 4800 baud
FCD6                   byte    "7200",LF,0            ;output parameter as 7200 baud
FCDC                   byte    "9600",LF,0            ;output parameter as 9600 baud
FCE2                   byte    "19200",LF,0           ;output parameter as 19200 baud
FCE9                   byte    0

FCEA           parms:
FCEA                   byte    "line ",0
FCF0                   byte    "ansi ",0
FCF6                   byte    "lfcr ",0
FCFC                   byte    "echo ",0
FD02                   byte    "hup  ",0
FD08                   byte    "tran ",0
FD0E                   byte    "tab  ",0
FD14                   byte    "edit ",0
FD1A                   byte    "map  ",0
FD20                   byte    "none",0
FD25                   byte    "even",0
FD2A                   byte    "odd ",0
FD2F                   byte    "any ",0
FD34           par.str:byte    " parity",LF,0

FD3D           delmsg: byte    "delay padding:  bs ht lf vt ff cr",LF,"                 ",0
FD70           brkmsg: byte    "break class characters:",LF,0
FD89           funcmsg:byte    LF,"cmd esc int str stp tnl rpl dll dlc dlw",LF,0

FDB3           speed.def:byte  $3E                    ;9600 baud
FDB4           parity.def:byte $4E                    ;8bit data, no parity
FDB5           parmdef:byte    0,4,0,0,0,1,1,0,0      ;lin,-ansi,lfcr,-ech,hup,-trans,-tab
                                                      ;edit,map
FDBE                   byte    0,0,0,0,0,0            ;padding for:  bs,ht,lf,vt,ff,cr

               ;TIE message area

FDC4           messdisc:byte   "Disconnected",LF,0
FDD2           conbroke:byte   "Connection broken",LF,0
FDE5           noconn: byte    "Cannot connect",LF,0
FDF5           alrcon: byte    "Already connected",LF,0
FE08           attmess:byte    LF,"Attention character is ",0
```

```
FE21               mess1:  byte    "Bad Address",LF,0
FE2E               mess2:  byte    "Bad Command",LF,0
FE3B               mess3:  byte    LF,"TIE: ",0
FE42               mess4:  byte    "Cannot connect to yourself",LF,0

FE5E               mess5:  byte    LF,"                Distant             Local"
FE88                       byte    LF,"TIE address        ",0

FE9D               mess7:  byte    "Connected",LF,0
FEA8               mess9:  byte    LF,"Remote disconnect",LF,0
FEBC               mess10: byte    "Bad parameter",LF,0

FECB               mess11: byte    LF,"                    Maximum         Total"
FEF5                       byte    LF,"collisions               ",0
FF10               mess12: byte    LF,"retransmissions          ",0
FF2B               mess13: byte    LF,"transmissions, incl. retrans.        ",0

FF50               delstr: byte    ESC,"[K",0      ;CSI/EL (control sequence introducer/erase to EOL)
FF54               escstr: byte    ESC,"[",0       ;CSI (ESC 5/11)
FF57               clstr:  byte    ESC,0           ;C1 control string (escape char)
FF59               bsstr:  byte    BS,SP,BS,0      ;back space string
```

```

            FFCO               org      $FFCO
     FFCO                term.def:
     FFCO              byte     $00,0                    ;end of function list
     FFC2              byte     $0A,1,$12,8              ;break class, repeat line
     FFC6              byte     $11,14,$13,12            ;start output, stop output
     FFCA              byte     $14,20,$1B,18            ;attention character, escape character
     FFCE              byte     $17,2,$15,6              ;delete word, delete line
     FFD2              byte     $0D,11,$08,4             ;new line and break class, delete char
     FFD6              byte     $03,16                   ;interrupt character
     FFD8        SRCaddr:byte     $04,$20                  ;address of this tie

     FFDA       backoffl:byte     5,30,60,120,240,480,960,1920
     FFE2              byte     2840,6860,17360,35720,54440,59000,64000,65000
     FFEA       backoffh:byte     5>8,30>8,60>8,120>8,240>8,480>8,960>8,1920>8
     FFF2              byte     2840>8,6860>8,17360>8,35720>8,54440>8,59000>8,64000>8,65000>8

```

```
                          bin

FFFA 03E0       nmivec: addr    reset           ; NMI vector
FFFC 03E0       resvec: addr    reset           ; Restart vector
FFFE 03E0       irqvec: addr    reset           ; IRQ vector
```

SYMBOL TABLE

```
.        FFFE    ANY      0003    BEL      0007    BREAK    0001    BS       0008    BUSY     0001    CDwait   E030
CNTLC    0003
CNTLD    0004    CNTLQ    0011    CNTLR    0012    CNTLS    0013    CNTLT    0014    CNTLU    0015    CNTLW    0017
COLMAX   000F
CONN     0004    CONNECT  0018    CONNstat 0290    CR       000D    CTTY     0085    DATA     0001    DCD      0040
DEFTAB   FF00
DISCON   0005    ENK      0005    ENQ      0082    EOT      0004    ESC      001B    ESCAPE   0007    EVEN     0001    FF
000C
FRerr    0020    GTTY     0084    HARDWD   1C00    HEADSIZ  0005    HT       0009    HUPflg   0066    IDLE     0000
INTR1    0080
INTR2    0081    LF       000A    LINSIZ   0050    NONE     0000    NOP      0000    NUL      0000    ODD      0002
PAGE0    0000
PAGE2    0200    PARM.MAX 0009    PROCMAX  0002    RCON     0010    RDISC    0002    REPT     0087    RPTSIZ   002A    RRb
0008
RTS      0080    SCON     0008    SDISC    0001    SP       0020    SRCaddr  FFD8    SSERV    0086    STTY     0083    SUB
001A
TIEaddrh 1800    TIEaddrl 1801    VT       000B    ackflag  0046    acktimer 0052    ackwait  EAC4    acr6522  C00B
addcount EDBF
addpar   E73E    adjy01   E78C    adjy10   E793    ai0cntr  1003    ai0data  1000    ai0mode  1002    ai0stat  1001
alrcon   FDF5
attmess  FE08    backoffh FFEA    backoffl FFDA    backup   E57B    badaddr  F479    badparm  F242    brkmsg   FD70
broken   ED1B
bsstr    FF59    buildbyt F4A4    clstr    FF57    cltab    F820    cansi    F20A    cany     F220    cbreak   F0CD
cchange  F00E
cconn    F3E9    ccooked  F1FE    cdelay   F322    cdiscon  F3D7    cecho    F1ED    cedit    F179    ceven    F22E
chkht    E727
chknet   E62D    chkstat  E6AB    chkterm  E291    chup     F15B    clfcr    F216    clrrb0   1C02    clrrb1   1C04
clrrb2   1C06
clrtrbfs E9E3    cmap     F183    cmdbuf   02AF    cmdbuf.1 02AE    cmdedit  007E    cmdindx  F004    cmdintrp EE57
cmdlist  EEF6
cmdmess  F302    cmdpntr  0297    cmdrout  EEB6    cmdstate 02AD    cmdstr   EFDB    cnosnsi  F20F    cnobreak F13D
cnoecho  F1F2
cnoedit  F17E    cnohup   F160    cnolfcr  F21B    cnomap   F188    cnone    F227    cnotab   F174    cnotrans F16A
cntltab  F800
cntrfnc  E456    codd     F235    colcnt   0040    conbroke FDD2    conmess  EDA5    conncnt  029B    contimer 02E1
counters 005F
cparsm   F249    crate    F1C1    craw     F1F7    csitab   F840    cstat    F18D    ctab     F16F    ctrans   F165
curbufs  0035
curproc  0003    curptype 029C    currbufh 0037    currbufl 0036    currbufx 004B    currcnt  004A    cursent  0042
cursub   0043
curtrbuf 0048    curtyp   004F    datarecd 02CC    datatrns 02C9    dbrecd   02D3    dbtrns   02CF    ddra6522 C003
ddrb6522 C002
decide   E53C    decque   EDB3    delays   006B    delchars EFC2    delmsg   FD3D    delstr   FF50    discstat 029E
dispat   E2DE
dispatch E69A    distaddh 0059    distaddl 0058    distimer 0051    done     ED15    drr      003E    dstaddr  0001
echo     E59F
echo.fre 0014    echo.inp 0012    echo.off 0065    echo.out 0013    echo.sta 0430    echo.use 0015    echoal   E5A7
edit     0069
enable   E084    entrysiz 0299    escpl    E555    escstr   FF54    esctab   E9D5    excsi    E6F3    fcdlc    0004
fcdll    0006
fcdlw    0002    fcesc    0012    fcint    0010    fcnop    0000    fcrpl    0008    fcstat   003B    fcstp    000C
fcstr    000E
fctask   003A    fcterm   E7A3    fctie    0014    fctnl    000A    ffbs     0010    ffcbt    0056    ffcha    0044
ffcht    0054
ffcnl    0040    ffcpl    0042    ffcr     000C    ffcsi    0008    ffctl    0004    ffcub    003E    ffcud    003A
ffcuf    003C
ffcup    0046    ffcuu    0038    ffdch    0030    ffdl     0032    ffech    002A    ffed     002E    ffel     002C
ffesc    0006
ffff     0012    ffhpa    001E    ffhpr    0020    ffht     000E    ffhts    0050    ffhvp    0026    ffich    0034
ffil     0036
```

```
ffind     0016  ffnl     000A  ffnop    0000  ffnp     004C  ffpld    001A  ffplu    001C  ffpp     004E
ffri      0018
ffsd      004A  ffsgr    0028  ffsu     0048  fftbc    0052  ffvpa    0022  ffvpr    0024  ffvt     0014
flstimer  02BE
fmnet     E7BF  fmnetsp  01C0  fmque    EB0F  fmterm   E2F2  fmtermr0 E3D5  fmtermrl E3D8  forcDISC EAF9
forcENQ   EB04
found     EE70  frmretn  E7E5  funcmsg  FD89  getchar  E2C2  getpack  E7EB  ginlen   0293  ginsav   0295
ginstat   0294
gonop     EA34  goreleas 029D  hardcnt  0000  hardwd   0000  hdreset  E000  hostconn 029A  hostsiz  EAB4
ier6522   C00E
ifr6522   C00D  inccount EDBD  inform   003D  init     E100  initALL  E104  initall  0002  inptab   007F
instatl   007B
interp    E68F  irqvec   FFFE  jobsp    0007  jobstat  0004  lastlost 02DB  lcol     001E  legal    E847
lfcr      0064
lookup    EE85  makASCI  F4C6  map      006A  mapchars EFD5  maxcol   02DF  maxlost  02DA  messl    FE21
mess10    FEBC
messll    FECB  mess12   FF10  mess13   FF2B  mess2    FE2E  mess3    FE3B  mess4    FE42  mess5    FE5E
mess7     FE9D
mess9     FEA8  messdisc FDC4  messpnt  005C  mnwait   EAF3  netread  0005  netstate 0296  nettab   E995
netwrite  0006
newrpnt   E7BA  nmivec   FFFA  noconn   FDE5  nomap    E73E  nooutput E685  normal   E314  nosucces 0047
notfound  EE7E
notsent   EBE4  nxtchar  EEA5  nxtindx  E2EA  nxtlst   EE90  ora6522  C001  orb6522  C000  outASCII E0C2
outaddr   E0B3
outmess   E79B  outmessh 005B  outmessl 005A  outnum2  F4B7  outspd   FC88  outstat  EDDD  outstatl 0063
outstat2  007D
cutterm   0004  packtyp  0005  par.str  FD34  parity.d FDB4  parml    0016  parm2    0017  parm3    0018
parm4     0019
parm5     001A  parm6    001B  parm7    001C  parm8    001D  parmdef  FDB5  parmindx 0292  parms    FCEA
partype   0079
pcr6522   C00C  pouts    003F  prntbrk  F0E8  prntchr  E22C  prntdel  F3AA  prompt   F8A0  putchar  E76D
putmess   E608
putout    E70A  que      0020  quecnt   001F  queinp   0028  queoutp  0029  quepack  E446  rawcook  0062  rbu
0291
rbuf.out  004E  rbufadh  EA0C  rbufcnt  0049  rbufouth 004D  rbufoutl 004C  rbufpnt  0055  rcv0buf  0500
rcvlbuf   0600
rcv2buf   0700  rdiscon  ED63  realtyp  0050  reccnt   002C  recinp   002A  recoutp  002B  recque   002D
rellret   E8D4
relall    E975  relbufh  0039  relbufl  0038  release  E3F3  relrbuf  EA11  relrbufl EA28  reltab   EA07
repcnt    EAA5
reptc0    E786
```

```
reptcl   E780   reset    E003   resvec   FFFC   return   E4C6   rnum     0044   rnumtab2 E7B6   rptaddr  0100
rrdy     0002   s4       E701   savstat  007A   savstat1 007C   sched    E2D0   sdiscon  ED36   sendisc  003C
sentDISC 029F
setclk   E287   setpar   F239   settim   EBD6   skip     F480   skipto   F496   skpterm  E1A2   spdconv  EFB6
spdlist  EF85
spdloop  E06D   spdmsg   EF77   spdsense E06B   spdtab   FC80   speed.de FDB3   srcaddr  0003   state0   E3AC
state1   E39B
state2   E390   state3   E323   stopinp  0057   stopoutp 0056   string   E5F5   strmess  F312   submit   EB35
subnop   EB3F
success  0040   t1h6522  C005   t1l6522  C004   t2f      0020   t2h6522  C009   t2l6522  C008   tab      0068
taback   EA9D
tabackx  EAA1   tabindx  0298   tabpoint 005D   tabs     02A1   tabseq   EA99   tabx     02A0   tcnt     0041
temp     02BF
temp1    02C0   temp4    02C1   tempesc  02C2   term.def FFC0   termpara 0062   termtab  F900   tiestat  02E4
timer    E24B
timinit  E007   tin.brk  000F   tin.buf  000B   tin.cnt  000E   tin.p    000D   tnum     0045   tonet    EACC
tonetsp  01A0
totcol   02DC   toterm   E5C0   totermr0 E707   totermsp 01E0   totlost  02D7   totrecd  02C6   tottrns  02C3
tout.buf 0010
tran.use 000A   transflg 0067   trbuf0   0200   trbuf0pt 00AA   trbuf1   0300   trbuf1pt 00AB   trbuf2   0400
trbuf2pt 00A4
trdy     0001   trstart  0006   wait     E242   waitecho ED9A   waitmess E60C   waitnet  ED8D   wdcount  02E0
wdtimcnt 0054
wdtimer  0053   x        0000   xbs      F9A0   xcdlc    E4C7   xcdll    E51E   xcdlw    E4CD   xcesc    E475
xcht     FA02
xcint    E4A0   xcnl     F9F0   xcnop    E497   xconndat E8DD   xconndis E8B0   xconnesc E93B   xcr      F98F
xcrpl    E4B1
xcsi     F967   xcstp    E492   xcstr    E46C   xctie    E47E   xctl     F95E   xctnl    E49B   xcub     F9DE
xcud     F9F8
xech     F9C7   xesc     F964   xgtty    E96D   xht      F9A8   xidlecon E8FD   xignore  E8D4   xint1    E94C
xint2    E954
xnl      F97F   xnop     F95B   xnrm     E6A3   xsconcon E92C   xshoit   F958   xstty    E95D   y        0001
zeropntr F998
```

CREF TABLE

```
.             0#
ANY         312#   3713
BEL          13#
BREAK       104#   3498   3479   3447   3425   1194
BS           14#   4497   4497   4381   4374   3324    683
BUSY        107#   4067   2314
CDwait      443#    449
CNTLC        25#
CNTLD        26#
CNTLQ        27#
CNTLR        28#
CNTLS        29#
CNTLT        30#
CNTLU        31#
CNTLW        32#
COLMAX      141#   2775   2522
CONN        122#   4070   2981   2936   2847   2646   2297   2169   2128
CONNECT     139#   3959   2962   2951   2643   2322   1246    950    878
CONNstat    330#   4068   3974   2315   2120   2075    851
CR           19#   4393   4316   4259   3324   1932   1897   1121
CTTY        130#   2721   2381
DATA        121#   2903   2806   2740   2657   2212   1289
DCD           9#    943    570    446
DEFTAB       40#    677    673
DISCON      123#   2924   2618
ENK          21#
ENQ         127#   2628   2177
EOT          22#
ESC          34#   4496   4495   4494   1638   1097
ESCAPE      124#   2735   2732   2172
EVEN        310#   3721    707
FF           18#   3324
FRerr         8#   1029    578
GTTY        129#
HARDWD       61#    435    433
HEADSIZ      86#   2743   2729   2681   2499   1732   1718   1282
HT           15#   4117   4105   3324   1908   1553   1444   1431
HUPflg      301#   3562   3556    954
IDLE        136#
INTR1       125#   1390
INTR2       126#   2354
LF           16#   4492   4491   4490   4489   4487   4486   4486   4485   4483   4482   4480   4479   4478   4477   4476   4475
           4474    4473   4472   4462   4462   4461   4460   4458   4441   4440   4439   4438   4437   4436   4435   4434   4433
           4432    4431   4430   4429   4428   4427   4426   4397   4320   4259   3952   3840   3629   3528   3521   3324   3298
           3176    1904   1891   1398   1387   1377   1126   1124    801
LINSIZ     4418#   1949   1234   1226
NONE        309#   3717    703
NOP         120#
NUL          12#   1940
ODD         311#   3725    709
PAGE0        65#    626    161
PAGE2        66#    723    614    328
PARM.MAX    145#   3758    696    694    692
```

```
PROCMAX    143#   990
RCON       138#  2959  2294
RDISC      111#  2928  2251
REPT       132#  3043  3016  2978  2921  2684  2649
RPTSIZ      88#  2714  2696  2694  2687
RRb        115#  2675  2497
RTS         10#  1963  1962
SCON       137#  2942
SDISC      110#  2931  2238
SP          23#  4497  4369  4115  4103  3946  3821  3813  3618  3511  3153  3132  1916  1800  1550  1442  1429
SRCaddr   4511#  4027  4024  3626  3624  2138  2133   764   757   530   528
SSERV      131#
STTY       128#
SUB         20#
TIEaddrh    57#   762
TIEaddrl    58#   769
VT          17#  3324
ackflag    255#  2830  2761  2600  2491  2243  2199
acktimer   273#  2954  2946  2888  2835  2822   844   841
ackwait   2578#  2821
acr6522     52#   425
addcount  3098#  1730  1278
addpar    1923#  1918  1905  1902
adjy01    1986#  4383  4376  4371
adjy10    1991#
ai0cntr      4#  3779  3727  3656  2030  1033   721   699   510   502   500   486   484
ai0data      1#  1966  1045  1035   581   563   525
ai0mode      3#  3781  3780  3728  3658  3657   700   507   505   491   489
ai0stat      2#  1961  1028   942   935   577   573   569   559   512   445
alrcon    4475#  3982  3981
attmess   4476#   796   795
backoffh  4515#  2766  2514
backoffl  4513#  2764  2512
backup    1542#  1468  1447  1434  1413
badaddr   4086#  4012  4000  3997  3991
badparm   3734#  3926  3914  3911  3909  3902  3900  3898  3896  3652  3640  3539  3422  3380  3374  3361
brkmsg    4461#  3491  3489
broken    2977#  2957  2866  2855  2776  2523
bsstr     4497#  1113  1111
buildbyt  4124#  4021  4018
clstr     4496#
cltab     4228#  1828
cansi     3689#  3250
cany      3712#  3253
cbreak    3469#  3247
cchange   3346#  3248
cconn     3974#  3246
ccooked   3679#  3250
cdelay    3869#  3247
cdiscon   3958#  3246
cecho     3663#  3250
cedit     3586#  3252
ceven     3720#  3253
chkht     1907#  1892
chknet    1697#  1623
chkstat   1817#  1798
chkterm    895#   839
chup      3555#  3250
cltcr     3701#  3250
```

```
clrrb0      74#  2391   753
clrrb1      75#  2392   754
clrrb2      76#  2393   755
clrtrbfs  2429#  3041  3014  2352  1383
cmap      3596#  3252
cmdbuf     363#  4128  4125  4113  4102  4098  4007  4005  3925  3918  3905  3639  3541  3476  3397  3231  1158  1135
cmdbuf.i   362#  3548  3538  3483  3473  3373  3369  1159  1146  1140  1131  1116  1108
cmdedit    320#  3438  1105   684
cmdindx   3339#  3401  3377
cmdintrp  3181#  1137
cmdlist   3259#  3189  3187
cmdmess   3845#  3768  3751
cmdpntr    345#  3871  3642  3348  3228  3185
cmdrout   3245#  3202  3200
cmdstate   361#  1358  1101  1092  1088
cmdstr    3327#  3859  3856  3355  3353
cnoansi   3693#  3245
cnobreak  3534#  3248
cnoecho   3669#  3245
cnoedit   3591#  3252
cnohup    3561#  3246
cnolfcr   3705#  3245
cnomap    3601#  3252
cnone     3716#  3253
cnotab    3581#  3251
cnotrans  3571#  3251
cntltab   4215#  1803
cntrfnc   1325#  1217  1215
codd      3724#  3253
colcnt     248#  2783  2778  2774  2763  2760  2530  2525  2521  2511  2506
conbroke  4473#  2989  2988
conmess   3074#  2968  2327
conncnt    350#  2854  2853
contimer   397#   881
counters   292#  3117  3115  3112  3110  3107  3105  3099   724
cparam    3740#  3248
crate     3636#  3246
craw      3675#  3245
csitsb    4240#  1870
cstat     3607#  3247
ctab      3576#  3251
ctrans    3566#  3251
curbufs    233#  2468  2467  2400  2279  2278  2072
curproc    173#   993   987   830   634
curptype   351#  2225  2211  2185  2168  2127  2118
currbufh   235#  2112
currbufl   234#  2373  2369  2337  2308  2301  2176  2158  2150  2145  2137  2132  2116
currbufx   263#  2479  2466  1708
currcnt    262#  3053  2398  1738  1727  1700
cursent    250#  3035  3008  2901  2899  2665
cursub     251#  2827  2667
curtrbuf   259#  2908  2906  2753   751
curtyp     269#  2977  2896  2838  2677  2640  2629  2619  2604
datarecd   388#  2285
datatrns   387#  2809
dbrecd     390#  1729
dbtrns     389#  1277
ddra6522    45#   421
```

```
ddrb6522    44#   418
decide    1486#  1458   1414
decque    3087#  2663   2653
delays     306#  3944   3931   1937
delchars  3315#  3878   3876
delmsg    4460#  3938   3936
delstr    4494#
discstat   354#  2932   2927   2252   2237
dispat    1001#
dispatch  1804#  1871   1829
distaddh   283#  4057   4032   4022   3611   2700   2693   2302   2144
distaddl   282#  4062   4055   4050   4042   4031   4026   4019   3613   2857   2703   2695   2309   2149
distimer   272#  3967   2591    963    875    872
done      2970#  2960   2952   2937   2934   2919   2897
drr        246#  2660   2166    731
dstaddr     91#  4065   4064   4063   4060   4059   4058   2860   2859   2858   2702   2699   2312   2311   2310   2305   2304   2303   2131
echo      1575#  1530   1528   1525   1523   1499   1493   1403   1399   1388   1386   1223
echo.fre   206#  1689   1688   1648   1593   1588    727
echo.inp   204#  1599   1591
echo.off   300#  3681   3670   3664   2724   2370   1579
echo.out   205#  1670   1635   1625
echo.sta   101#  1664   1626   1592
echo.use   207#  3516   3064   1691   1690   1649   1622   1594
echoal    1584#  4159   3953   3947   3939   3937   3862   3860   3852   3850   3841   3823   3822   3815   3814   3801   3799
          3795   3793   3777   3775   3772   3770   3748   3630   3620   3529   3522   3512   3510   3506   3492   3490   3177
          3154   3133   2005   2003   1125   1114   1112    818    813    802
edit       304#  3833   3592   3587   1205
enable     509#   493
entrysiz   347#  3874   3351   3221   3182
escpl     1508#  1497   1490
escstr    4495#  1524   1522
esctab    2422#  2345   2343
excsi     1865#  1839
fcdlc      149#  3434   3339
fcdll      150#  3339
fcdlw      148#  3339
fcesc      156#  3340
fcint      155#  3340
fcnop      147#
fcrpl      151#  3830   3339
fcstat     242#   730
fcstp      153#  3340
fcstr      154#  3341
fctask     240#  3079   2915   2444   2034   1617   1265
fcterm    2018#  1620
fctle      157#  3803   3379   3341    786
fctnl      152#  3443   3340
ffbs      4177#  4218
ffcbt     4212#  4247
ffcha     4203#  4242
ffcht     4211#  4243
ffcnl     4201#  4242
ffcpl     4202#  4242
ffcr      4175#  4219
ffcsi     4173#  4234
ffctl     4171#  4223   4223   4223   4223   4222   4222   4222   4221   4221   4221   4221   4220   4220   4220   4220   4219
          4219   4219   4218   4218   4217   4217   4217   4217   4216   4216   4216   4216
ffcub     4200#  4242
```

```
ffcud      4198#  4241
ffcuf      4199#  4241
ffcup      4204#  4243
ffcuu      4197#  4241
ffdch      4193#  4245
ffdl       4194#  4244
ffech      4190#  4247
ffed       4192#  4243
ffel       4191#  4243
ffesc      4172#  4222
ffff       4178#
ffhpa      4184#  4249
ffhpr      4185#  4249
ffht       4176#  4218
ffhts      4209#  4230
ffhvp      4188#  4250
ffich      4195#  4241
ffil       4196#  4244
ffind      4180#  4229
ffnl       4174#  4229
ffnop      4170#  4252   4252   4252   4251   4251   4251   4251   4249   4249   4248   4248   4248   4248   4247   4247   4246
           4245   4245   4244   4244   4235   4235   4235   4235   4234   4234   4234   4233   4233   4233   4233   4232   4232
           4232   4232   4231   4231   4230   4230   4229   4229   4228   4228   4228   4228
ffnp       4207#  4246
ffpld      4182#  4230
ffplu      4183#  4231
ffpp       4208#  4246
ffri       4181#  4231
ffsd       4206#  4246
ffsgr      4189#  4252
ffsu       4205#  4245
fftbc      4210#  4250
ffvpa      4186#  4250
ffvpr      4187#  4250
ffvt       4179#
flstimer    365#   870    862    734
fmnet      2070#  3020   2363   2289   2266   2095    651    649
fmnetsp      37#   653    652    650
fmque      2638#  2595
fmterm     1027#   973
fmtermr0   1223#  1378   1373   1211   1201   1184   1177   1170   1167
fmtermrl   1224#
forcDISC   2618#  2592   2589
forcENQ    2628#  2598
found      3197#
frmretn    2094#  2086   2079   2076
funcmsg    4462#  3800   3798
getchar     971#   937
getpack    2102#  2108   2073
ginlen      337#
ginsav      339#
ginstat     338#
gonop      2490#  2833
goreleas    352#  2432   1297   1253    975
hardcnt      90#  2744   2730   2688   2682   2500   1715   1283
hardwd      164#  1004    567    556    518    464    443    436    434
hdreset     407#  3247    951    407
hostconn    349#  4054   2850
hostsiz    2576#  4053
```

```
ier6522     55#  415
ifr6522     54#  837   466
inccount  3097# 2871  2810  2800  2781  2545  2528  2286  2155   882
inform     244# 3039  3012  2632  2597  2474  2406  2089
init       604# 2979  2922
initALL    607#  579
initall    168#  668   618   605   411
inptab     324# 3806  3545  3542  3496  3480  3477  3448  3446  3431  3429  3426  3424  3420  3405  3393  3383
          1203  1193  1179   784   678
instatl    316# 3683  2375  1355  1352  1343  1340  1169  1130  1075  1039
interp    1797# 1750  1681  1645
irqvec    4522#
jobsp      189# 1001   832   661   654   647
jobstat    180#  991
lastlost   393# 2894  2891
lcol       219# 4385  4354  4348  4341  4335  1954  1948  1899  1810
legal     2154# 2141
lfcr       299# 3707  1894
lookup    3214# 3881  3649  3358  3192
makASCI   4150# 4144
map        305# 3602  3597  1888
mapchars  3324# 3943  3929
maxcol     395# 3127  2786  2784  2534  2531
maxlost    392# 3148  2890  2876  2874
mess1     4477# 4091  4090
mess10    4487# 3735  3734
mess11    4489# 3124  3123
mess12    4491# 3145  3144
mess13    4492# 3166  3165
mess2     4478# 3208  3207
mess3     4479# 1349  1348
mess4     4480# 4035  4034
mess5     4482# 3608  3607
mess7     4485# 3075  3074
mess9     4486# 3026  3025
messdisc  4472# 2998  2997
messpnt    288# 1683  1675  1673
mnwait    2613# 2655  2608
netread    182#  733
netstate   341# 3958  2963  2950  2943  2939  2642  2323  2295  2224  1245   949   877
nettab    2413# 2230  2228
netwrite   183# 3964  2611  2475  2407  2256  2198  2090  1313   967
newrpnt   2064# 2106
nmivec    4520#
noconn    4474# 2985  2984
nomap     1922# 1912  1909  1895  1889
nooutput  1776# 1704  1698
normal    1075#
nosucces   256# 2797  2757  2542  2509  2244
notfound  3207# 3195
notsent   2769# 2794
nxtchar   3229# 3235
nxtindx   1018# 3089  2274  1711  1318
nxtlst    3219# 3232
ora6522     42# 2771  2518  2070
orb6522     41# 2754  2503   867   865   718   431
outASCII   543#  539
outaddr    533#  531   529
[illegible] 2003# 4092  4036  3983  3736  3609  3209  3167  3146  3125  3076  3027  3000  1350   797
```

```
outmessh    287#  1659
outmessl    286#  1676  1665
outnum2    4138#  3945  3627  3625  3614  3612  3174  3172  3170  3163  3161  3159  3149  3142  3140  3138  3128
outspd     4425#  3792  3784
outstat    3123#  3632  3029  3002
outstat1    298#  3695  1751  1749
outstat2    319#  3684  1682  1680  1646  1644
outterm     181#  3080  2916  2445  2283  1777  1601  1335  1266
packtyp      93#  2750  2746  2734  2727  2725  2501  2367  2336  2175  2157  2115
par.str    4458#  3771  3769
parity.d   4465#   504   488
parm1       209#  4411  4409  4406  4398  4382  4375  4370  4302  1861  1857  1851
parm2       210#  4303
parm3       211#  4304
parm4       212#  4305
parm5       213#  4306
parm6       214#  4307
parm7       215#  4308
parm8       216#  4309
parmdef    4466#   688
parmindx    335#  4310  1850  1845
parms      4444#  3849  3846
partype     307#  3761  3729  1923   710
pcr6522      53#   428
pouts       247#  2886  2813  2217
prntbrk    3488#  3474
prntchr     808#  3819   800
prntdel    3935#  3884
prompt     4259#   521
putchar    1961#  1970  1941  1930
putmess    1672#
putout     1887#  4395  4384  4363  4342  4328  4321  4318  4291  4285  1982  1976  1874  1811  1641
que         222#  2639  1316
quecnt      221#  3087  2607  2594  2588  2434  1317
queinp      223#  2435  1319  1315
queoutp     224#  3090  3088  2638  2436
quepack    1312#  4071  3044  3017  2382  2355  2298  1391  1290
rawcook     297#  3836  3682  2726  2374  1255  1238
rbu         334#
rbuf.out    267#  1737  1735  1733
rbufadh    2464#  2111  1709
rbufcnt     260#  2673  2494  2478  2470  2403  2215   635
rbufouth    266#  1710
rbufoutl    265#  1736  1716
rbufpnt     277#  2482  2272  2110  2107  2105  2102   732
rcv0buf      78#
rcv1buf      79#
rcv2buf      80#
rdiscon    3022#  2929
realtyp     270#  2803  2739  2736  2733  2678
reccnt      228#  3052  2399  2276  1713  1703
recinp      226#  2396  2275  2271
recoutp     227#  2397  1712  1706
recque      229#  2273  1707
rellret    2262#  2384  2377  2331  2317  2205  2186
relall     2391#  3032  3005  2360
relbufh     238#   714
relbufl     237#  2486  2484
release    1245#  1239  1235  1227   978
```

```
relrbuf    2466#  1743   1721
relrbufl   2482#  2263   2241
reltab     2463#  2483
repcnt     2573#  2865
reptc0     1978#  4377   4372
reptcl     1974#  4399
reset       410#  4522   4521   4520
resvec     4521#
return     1408#  1421   1396
rnum        253#  3036   3009   2670   2492   2209   2203
rnumtab2   2062#  2900   2208
rptaddr      63#  2701   2698
rrdy          7#   936    574
s4         1873#  1832
savstat     315#  3696   1875   1862   1846   1797
savstatl    317#  1353   1341   1168   1129
sched       987#   976    969    960
sdiscon    2994#  2245
sendisc     243#  3963   2622   2585   2255    966
sentDISC    356#   968    959
setclk      889#   885    804    791
setpar     3727#  3722   3718   3714
settim     2763#  2788
skip       4096#  3986   3888   3870   3637   3347
skipto     4113#  4096   3535   3470   3365
skpterm     713#   669
spdconv    3313#  3655
spdlist    3299#  3646   3644
spdloop     497#   582
spdmsg     3298#  3776   3774
spdsense    495#   498
spdtab     4424#   506
speed.de   4464#   490
srcaddr      92#  2300   2143    767    766    765    760    759    758
state0     1190#  1076
statel     1175#  1079
state2     1165#  1082
state3     1087#
stopinp     280#  2430   1581   1296   1262
stopoutp    279#  2431   1697   1366   1333   1141
string     1659#  1627
strmess    3855#
submit     2665#  2878   2862   2634   2624
subnop     2670#  2605
success      72#  2793   2538
tlh6522      48#  2767   2515
tll6522      47#  2765   2513
t2f          70#   838    467
t2h6522      50#   892    462
t2l6522      49#   890    461
tab         303#  3582   3577   1911
taback     2570#  2671   2493
tabackx    2571#  2666
tabindx     346#  3928   3654   3400   3376   3224   3216   3197
tabpoint    290#  3879   3877   3647   3645   3356   3354   3229   3225   3190   3188
tabs        359#  4360   1565
tab[illegible]     2569#  2668   2204
[illegible]         358#  4361   4359   4336   1955   1900   1564   1563   1477   1295
tcnt        249#  3038   3011   2887   2873   2868   2864   2820   2690
```

```
temp        368#   4146   4138   4133   4126   3930   3921   3915   3913   3903   3899   3892   3756   3744   3523   3507   3501
           3497    3437   3428   3419   3417   3398   3222   3219   3215   3106   3098   1860   1849   1496   1489   1452   1446
           1439    1433    815    811    799    793    540    534
templ       369#   4357   4349   3825   3808   3804   3621   3617   3433   3430   3407   3402   3385   3378   3155   3152   3134
           3131    1919   1915   1478   1472   1471   1463   1457   1451   1450   1438   1437   1424
temp4       370#   3949   3948   3942   1946   1935
tempesc     371#
term.def   4503#
termpara    295#   3745    689    624
termtab    4268#   1806   1804
tiestat     398#
timer       836#
timinit     413#    571
tin.brk     199#   2966   2439   2326   1294   1237   1195
tin.buf     196#   2453   1548   1454   1428   1402   1287   1285   1284   1230    748    744
tin.cnt     198#   2438   1546   1542   1465   1420   1395   1293   1279   1275   1258   1233   1232   1225
tin.p       197#   2448   1547   1545   1425   1405   1300   1231   1229    740
tnum        254#   3037   3010   2826   2162
tonet      2581#   3047   2971   2614    658    656
tonetsp      38#    660    659    657
totcol      394#   3141   3139   3137   2780   2527
toterm     1617#   1780   1774   1724   1693   1652    644    642
totermr0   1879#   1866   1844   1837   1826   1823
totermsp     36#    646    645    643
totlost     391#   3162   3160   3158   2870
totrecd     386#   2154
tottrns     384#   3173   3171   3169   2799   2709   2544
tout.buf    202#   2912   2910   2751   2452    747    743
tran.use    194#   2918   2441   1274   1249
transflg    302#   3796   3572   3567   1197   1176   1166
trbuf0       96#   4063   4058   2858   2310   2303    765    758    746    742
trbuf0pt     82#    750
trbufl       97#   4064   4059   2859   2311   2304    766    759
trbuflpt     83#
trbuf2       98#   4065   4060   2860   2746   2744   2734   2730   2727   2725   2710   2702   2699   2696   2694   2688   2682
           2501    2500   2312   2305    767    760
trbuf2pt     84#   2747   2502
trdy          6#   1963   1962    560    513
trstart      94#   2447   1401   1299    739
wait        826#   3067   3057   3019   2970   2948   2824   2769   2613   2517   2362   2288   2265   2248   2220   2094   1969
           1779    1773   1723   1685   1651
waitecho   3064#   3068   3030   3003
waitmess   1675#   1686
waitnet    3052#   3058   3022   2994
wdcount     396#   2092
wdtimcnt    275#   2183   2081
wdtimer     274#   2181   2085   2078    887    884
x             0#   4053   3943   3929   3745   3655   3545   3542   3496   3480   3477   3420   3401   3377   3231   3202   3200
           3089    2900   2821   2766   2764   2709   2639   2514   2512   2483   2345   2343   2274   2273   2230   2228   2111
           2106    1937   1870   1861   1857   1851   1828   1803   1711   1707   1664   1626   1592   1565   1318   1316   1203
           1193    1179   1158   1135    784    688    677    673    626    614    506
xbs        4341#   4270
xcdlc      1413#   1326
xcdll      1462#   1326
xcdlw      1420#   1326
xcesc      1340#   1328
xcht       4405#   4278
xcint      1383#   1328
xcnl       4393#   4276
```

```
xcnop     1372#  1326
xconndat  2271#  2419
xconndis  2237#  2420
xconnesc  2336#  2420
xcr       4327#  4269
xcrpl     1395#  1327
xcsi      4301#  4269
xcstp     1365#  1327
xcstr     1332#  1327
xctie     1348#  1328   1042
xctl      4291#  4268
xctnl     1377#  1327
xcub      4381#  4275
xcud      4397#  4275
xech      4369#  4273
xesc      4297#  4268
xgtty     2381#  2423
xht       4348#  4410   4269
xidlecon  2294#  2414
xignore   2263#  2423   2423   2422   2420   2420   2419   2419   2419   2418   2418   2418   2418   2417   2417   2417   2417
          2416   2416   2416   2415   2415   2415   2415   2414   2414   2414   2413   2413   2413   2413   2151   2146   2139
          2134   2129
xintl     2352#  2422
xint2     2360#  2422
xnl       4316#  4269
xnop      4287#  4278   4278   4278   4277   4277   4277   4277   4276   4276   4276   4275   4275   4274   4274   4274   4274
          4273   4273   4273   4272   4272   4272   4272   4271   4271   4271   4271   4270   4270   4270   4268
xnrm      1810#  1801
xsconcon  2322#  2416
xshoit    4285#  4268
xstty     2367#  2422
y            0#  4360   4128   4125   4113   4102   4098   4007   4005   3944   3931   3925   3918   3905   3806   3639   3541
          3476   3448   3446   3431   3429   3426   3424   3405   3397   3393   3383   3229   3225   3117   3115   3112   3110
          3107   3105   2865   2751   2710   2671   2668   2666   2493   2486   2373   2369   2337   2308   2301   2208   2204
          2176   2158   2150   2145   2137   2132   2116   1806   1804   1736   1716   1709   1676   1548   1454   1428   1402
          1284   1230   1217   1215   1004   1001    991    832    689    678    567    556    521    518    464    443
zeropntr  4334#  4394   4327   4317
```

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET *(See instructions)*	1. PUBLICATION OR REPORT NO. NBSIR 84-2902	2. Performing Organ. Report No.	3. Publication Date September 1984

4. TITLE AND SUBTITLE

Selected NBSNET Software

5. AUTHOR(S)

Michael Strawbridge, Sheryl Schooley, Robert Crosson

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

N/A

8. Type of Report & Period Covered

Interim

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

10. SUPPLEMENTARY NOTES

□ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)*

NBSNET is a local area communications network at the National Bureau of Standards. Ethernet-like in its design, it has operated successfully since 1979, supporting terminal-computer and computer-computer communications. Devices physically connect to NBSNET through RS-232-C interfaces; each being customized to the device being served. Customization primarily involves modifying the control program, called a "personality", for each interface. Each personality is divided into modules which implement, among other things, the network's internal protocol and the external device communications protocol. Three external device protocols are used. A listing of some typical personality modules is supplied.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

Communications; computer; LAN; NBSNET; network; protocol

13. AVAILABILITY

☒ Unlimited
□ For Official Distribution. Do Not Release to NTIS
□ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

112

15. Price

$13.00