```
EEEEEE  H     H     AA     H     H  N     N
E       H     H    A  A    H     H  NN    N
EEEEE   HHHHHHH   A    A   HHHHHHH  N N   N
E       H     H  AAAAAA   H     H  N  N  N
E       H     H  A    A   H     H  N   NN
EEEEEE  H     H  A    A   H     H  N    N

Wed Oct 27 16:50:49 1982

Listing file .../v4.lst for Eric Hahn at bbnx.

Listing file .../v4.lst for Eric Hahn at bbnx.

Listing file .../v4.lst for Eric Hahn at bbnx.

Wed Oct 27 16:50:49 1982

EEEEEE  H     H     AA     H     H  N     N
E       H     H    A  A    H     H  NN    N
EEEEE   HHHHHHH   A    A   HHHHHHH  N N   N
E       H     H  AAAAAA   H     H  N  N  N
E       H     H  A    A   H     H  N   NN
EEEEEE  H     H  A    A   H     H  N    N
```

```
EEEEEE  H    H    AA    H    H  N    N
E       H    H   A  A   H    H  NN   N
EEEEE   HHHHHH  A    A  HHHHHH  N N  N
E       H    H  AAAAAA  H    H  N  N N
E       H    H  A    A  H    H  N   NN
EEEEEE  H    H  A    A  H    H  N    N

Wed Oct 27 16:50:49 1982

Listing file .../v4.lst for Eric Hahn at bbnx.

Listing file .../v4.lst for Eric Hahn at bbnx.

Listing file .../v4.lst for Eric Hahn at bbnx.

Wed Oct 27 16:50:49 1982

EEEEEE  H    H    AA    H    H  N    N
E       H    H   A  A   H    H  NN   N
EEEEE   HHHHHH  A    A  HHHHHH  N N  N
E       H    H  AAAAAA  H    H  N  N N
E       H    H  A    A  H    H  N   NN
EEEEEE  H    H  A    A  H    H  N    N
```

```
    1 000F              nme:    equ 15
    2 0001              z80as:  equ 1
    3 0000              caz80:  equ 0

    0 0000              *include mvr.z80
                        ;network size

    3 0010              nbox:   equ 16              ;number of boxes per net
    4 0020              nlink:  equ 32              ;number of links per bo

                        ;MUNV is the version number of the communication mechani
                        ;anything in this file, update MUNS so you'll get error
                        ;skewed.

   10 0000              munv:   equ 0               ;this value will be in
   11 0004              mpunv:  equ 4               ;address of MPC's MUNV w
   12 F006              runv:   equ 0f006h          ;address of VROM's MUNV
   13 F003              rdie:   equ 0f003h          ;call VROM to crash
   14 2000              mwind:  equ 2000h           ;start address of MPC wi
   15 2200              msave:  equ 2200h           ;start address of MPC sa
   16 F000              vrom:   equ 0f000h          ;start address of VROM
   17 4000              rram:   equ 04000h          ;start address of RRAM (
   18 4200              vram:   equ 04200h          ;start address of VRAM

                        ;Commands in MVC:
   21 0000                                          ;"actions"
   22 0000              mnull:  equ 0               ;no command - window a
   23 0001              mrt:    equ 1               ;read text
   24 0002              mrp:    equ 2               ;read packet
   25 0003              mwt:    equ 3               ;write text
   26 0004              mwp:    equ 4               ;write packet
   27 0005              mer:    equ 5               ;end read operation
   28 0006              mew:    equ 6               ;end write operation
   29 0007              mkl:    equ 7               ;master clear (kill) lin
   30 0008              mrn:    equ 8               ;restart no-op (used by
   31 0009              mcli:   equ 9               ;clrbfi
   32 000A              mclo:   equ 10              ;clrbfo
   33 000B              msbk:   equ 11              ;send a BREAK on the lin
   34 000C              mr3:    equ 12              ;read KCNT, PCNT, statu

   36 0000                                          ;"read/write"
   37 000D              mrbaud: equ 13              ;baud rate (FF=hunt)
   38 000E              mwbaud: equ 14
   39 000F              mrecho: equ 15              ;local echo (0=no, 1=yes
   40 0010              mwecho: equ 16
   41 0011              mrprty: equ 17              ;parity (0=none, 1=odd,
   42 0012              mwprty: equ 18
   43 0013              mrpx:   equ 19              ;local processing of XO
   44 0014              mwpx:   equ 20
   45 0015              mrgx:   equ 21              ;local generation of XON
   46 0016              mwgx:   equ 22
   47 0017              mrclip: equ 23              ;GX clip value for XOFF
   48 0018              mwclip: equ 24              ; end of the fifo to sen
   49 0019              mrcbk:  equ 25              ;read and clear break a
   50 001A              mdial:  equ 26              ;"dial" out of network
   51 001B              mrpad:  equ 27              ;padding control
   52 001C              mwpad:  equ 28
   53 001D              mrint:  equ 29              ;Interrupt control (0=no
```

```
54 001E                 mwint:  equ 30          ;(does not affect break
55 0000
```

```
                        ;Responses.  Always negative.  40h bit says it's fatal.

  58 0080               merr:   equ 80h
  59 0006               mftbit: equ 6                   ;fatal bit (40h).
  60 00C0               mcerr:  equ merr+40h

  62 00C0               mcmder: equ mcerr+0             ;unknown command
  63 00C1               mtyper: equ mcerr+1             ;type mismatch (V-link
  64 0082               mnrmer: equ merr+2              ;no room for this output
  65 00C3               mliner: equ mcerr+3             ;bad value in MVL
  66 0084               mmtyer: equ merr+4              ;no data to be read
  67 0085               mrster: equ merr+5              ;MPC has restarted recen
  68 00C6               mdieer: equ mcerr+6             ;MPC is dead (in DIE loo
  69 00C7               marger: equ mcerr+7             ;some bad arg

                        ;Memory layout for window

  73 2000                       org mwind
  74 2000               mvc:    defs 1                  ;command/response word
  75 2001               mvl:    defs 1                  ;line number (0-3, use

  77 2002               mv1:    defs 1                  ; #1 general purpose arg
  78 2003               mv2:    defs 1                  ; #2
  79 2004               mv3:    defs 1                  ; #3

  81 0007               gonbit: equ 7                   ;call gone away (tracks
  82 0080               gonmsk: equ 128

  84 0006               carbit: equ 6                   ;carrier established (t
  85 0040               carmsk: equ 64

  87 0000               brkbit: equ 0                   ;break detected
  88 0001               brkmsk: equ 1

  90 2005               mvx1a:  defs 2                  ;transfer part 1 addres
  91 2007               mvx1c:  defs 2                  ;transfer part 1 count
  92 2009               mvx2a:  defs 2                  ;transfer part 2 address
  93 2008               mvx2c:  defs 2                  ;transfer part 2 count

  95 0050               clockv: equ 80                  ;2.4576MHZ/256/cps=clock
  96 200D                                               ;Where cps=number of in
  97 200D                                               ;clockv=80 = 120/sec whi

  99 001E               clocks: equ 30                  ;Number of CLOCKV rate
 100 200D                                               ;clock interrupt, 120/30

 102 200D               mvtime: defs 1                  ;1/120 second tick coun
 103 200D               mvend:  equ mvtime              ;end of the window area
 104 200E
```

```
                        ;The PBRG definitions

107 00FF                bhunt:  equ 0FFh        ;needed for CRATE macro
108 0000                b50:    equ 0
109 0001                b75:    equ 1
110 0002                b110:   equ 2
111 0003                b134p5: equ 3
112 0004                b150:   equ 4
113 0005                b300:   equ 5
114 0006                b600:   equ 6
115 0007                b1200:  equ 7
116 0008                b1800:  equ 8
117 0009                b2000:  equ 9
118 000A                b2400:  equ 0ah
119 000B                b3600:  equ 0bh
120 000C                b4800:  equ 0ch
121 000D                b7200:  equ 0dh
122 000E                b9600:  equ 0eh
123 000F                b19200: equ 0fh

                        ;IO ports, etc.

127 00F0                mpc0:   equ 0f0h        ;address of first MPC
128 00F7                mpc7:   equ 0f7h        ;address of last MPC
129 00FF                mpc99:  equ 0ffh        ;address of a non-exista
130 00FF                select: equ 0ffh        ;selection port

132 00FE                lights: equ 0feh        ;light socket.

134 0080                vtnup:  equ 80h         ;VTN is up.
135 0020                pkther: equ 20h         ;packet for another node
136 0010                pktme:  equ 10h         ;packet for me.
137 0008                iopw:   equ 08h         ;MPC I/O wait.
138 0004                buffw:  equ 04h         ;wait for buffer.
139 0002                pplmty: equ 02h         ;packet processing list

                        ;ASCII definitions

143 007F                low7:   equ 7fh         ;mask to strip possible
144 0001                ctrla:  equ 'A'-'@'     ;control-A for broadcast
145 0003                ctrlc:  equ 'C'-'@'     ;control-C for autoconn
146 0007                ctrlg:  equ 'G'-'@'     ;bell
147 0008                ctrlh:  equ 'H'-'@'     ;backspace
148 000A                ctrlj:  equ 'J'-'@'     ;line feed
149 000D                ctrlm:  equ 'M'-'@'     ;carriage return
150 0011                ctrlq:  equ 'Q'-'@'     ;control-Q (XON)
151 0013                ctrls:  equ 'S'-'@'     ;control-S (XOFF)
152 0018                ctrlx:  equ 'X'-'@'     ;control-X
153 007F                rubout: equ low7
154 200E
```

```
155 4000                        org rram
156 4000                        defs 3

                        ;Variable storage for VROM

160 4003                vdie:   defs 1          ;die reason
161 4004                vdiepc: defs 2          ;PC
162 4006                vdiesp: defs 2
163 4008                vdieaf: defs 2
164 400A                vdiebc: defs 2
165 400C                vdiede: defs 2
166 400E                vdiehl: defs 2
167 4010                vdieix: defs 2
168 4012                vdieiy: defs 2
169 4014
```

```
                        ;Hexadecimal record read area.

 172 4014               record:
 173 4014               rsize:  defs 1
 174 4015               raddr:  defs 2
 175 4017               rtype:  defs 1
 176 4018               rdata:  defs 64

 178 4058               riompc: defs 1
 179 4059               riolin: defs 1

 181 405A               rmpc:   defs 1          ;MPC i/o port during VRO

 183 405B               richc:  defs 1          ;count of buffered char
 184 405C               richa:  defs 2          ;address of buffered cha
 185 405E               richb:  defs 128        ;first half of buffer sp
 186 40DE               richb2: defs 128        ;second half of buffer

 188 415E                       defs 31
 189 417D               rstack: defs 1
 190 417E               rtimem: defs 1          ;address of reliable MPC
 191 417F               richt:  defs 1          ;tick time elapsed.
 192 4180               richs:  defs 1          ;second time elapsed.
 193 4181               richlt: defs 1          ;last read MVTIME.
 194 4182               patrn:  defs 1          ;light pattern.
 195 4183
```

```
                        ;Local Modes:
                        ;Mode:Macro
                        ;Comment Column:32
                        ;Comment Start:;
                        ;Auto Fill Mode:-1
                        ;Fill Column:79
                        ;End:
    0 4183              *include v.z80
                        ;VTN system, (C) Copyright 1979,1980,1981,1982 E. Hahn

    3 0000              major:  equ 0                   ;VTN major version numb
    4 0004              minor:  equ 4                   ;VTN minor version numbe

                        ;Edit History.  Please keep this formated and up-to-da
                        ;
                        ; Vers  Date        Who
                        ;
                        ; 0.0   22-Jul-82   WPI
                        ;
                        ;       Initial version received at BBN from WPI sourc
                        ;
                        ; 0.1   24-Jul-82   EAH
                        ;
                        ;       Removed "REAL" assembly flag.  Installed new end
                        ;       discussion thereof.  Added CNAME, version number
                        ;       Moved routing updates to front of PPL.  Added
                        ;       MPC padding.  Converted souce for UNIX assembly.
                        ;       LWATCH and !WATCH command.  Split out config stu
                        ;       buffer counters (FRLENG, etc.) be double-preci
                        ;       they overflowed on 32K+ machines.  Added patch s
                        ;       support for transparent breaks, interrupt charac
                        ;       SET INTERRUPT command.
                        ;
                        ; 0.2   10-Aug-82   EAH
                        ;
                        ;       Must be disconnected to do XCL commands, connect
                        ;       Integrated version 0.2 MPC and debugged Set Inte
                        ;       Set Delay with it.
                        ;
                        ; 0.3   24-Aug-82   EAH
                        ;
                        ;       Re-vamped modem control to conform to MPC 0.3 st
                        ;       Broadcast logic (it was buggy).
                        ;
                        ; 0.4   1-Sep-82    EAH
                        ;
                        ;       Fixed RNGBIT bug in CARDRP.  Added DECIN overfl
                        ;       Long RCall bug causing buffer counts to be wrong
                        ;       and re-organized LBLK.  Added new SET/!SET mecha
                        ;       End-End less anxious to call SYNC.  Changed REL
                        ;       PTXNCH to help long-delay connections.  Made spa
                        ;       to the command decoder.  Made !Watch report node
                        ;       now and then.  Fixed packet pictures (DATA and
                        ;       Split out 0.Z80 to have XCL commands, etc.  Adde
                        ;       to interlock XCL commands.  Cleaned up DEAD stat
                        ;       bugs in transparent-break.  Made it legal to co
                        ;       (undid part of ver 0.2).  Removed List command,
```

```
;       Removed LOWMPC, always use MPCO.  Added MPC sele
;       diagnostics at IMPC and made init run through a
;       before starting VTN.  Made command decoder bee
;       command errors.  Added ABORTP and ABORT.
```

54 4183

```
                        ;Known bugs/deficiencies in VTN (flagged && in comments)

 57 4183                ;VTN should be able to make RING transition a fe
 58 4183                ;on-behalf of connections
 59 4183                ;48K
 60 4183                ;VLINKs should take a while to come back up
 61 4183                ;Add LED support for crashes (also RR)
 62 4183                ;Add stack guard word checked by ITRAN
 63 4183                ;Re-do VTN-side modem control, make 2-stage conn
 64 4183                ;Routing packets should go only to highest # VBO
 65 4183                ;Baud rate matching in connect command (hard!)
 66 4183                ;CAMP-ON connections
 67 4183                ;Interleave MPCs and line block scanning
 68 4183                ;Permanent Virtual Circuits, Initial Virtual Ci
 69 4183                ;Start numbers at 1, not zero (hosts, nodes, lin
 70 4183
```

```
                        ;Error numbers:

    73 4183
```

```
                        ;VTN Parameters

76 007F                 pmax:   equ 127                 ;maximum packet size

                        ;Debugging features. 0=don't include code to check it,1
                        ;code which is used for debugging is nominally flagged w

81 0001                 debbuf: equ 1                   ;watch for non-buffers
82 0040                 vpat:   equ 64                  ;patch space (bytes) at

                        ;Timers in 1/2 second units

86 0002                 ruptim: equ 2                   ;1 second between RUPs
87 0002                 rupnum: equ 2                   ;2 missing will declare
88 00B4                 cmdtim: equ 90*2                ;90 seconds will kill id
89 000A                 rngtim: equ 5*2                 ;5 seconds for carrier t
90 0004                 dattim: equ 2*2                 ;2 seconds to clear MPC
91 4183                                                 ; has been dropped by li
92 4183
```

```
                        ;VTN Line block

   95 0000                      org 0

   97 0080              lsize:  equ 128             ;line block size rounde

   99 0000              ldbeg:                      ;start of DBLK image por
  100 0000              ltype1: defs 1              ;VTN line description (
  101 0001  0100        ltype2: defw 1              ;VTN line flow control (
  102 0003              lclip:  defs 1              ;MPC keyboard clip
  103 0004              lhost:  defs 1              ;if HBIT=1, host number
  104 0005              lautoh: defs 1              ;auto-connect hi byte
  105 0006              lautol: defs 1              ;auto-connect lo byte
  106 0007              logrp:  defs 1              ;originate group
  107 0008              lagrp:  defs 1              ;answer group
  108 0009              lprty:  defs 1              ;parity.
  109 000A              lrate:  defs 1              ;baud rate
  110 000B              lpad:   defs 1              ;padding
  111 000C              lintr:  defs 1              ;interrupt character
  112 000C              ldend:  defl $-1            ;end of DBLK image

  114 000D              lnum:   defs 1              ;link number (0-31)
  115 000E              lmymsk: defs 1              ;mask for status word
  116 000F              lsp:    defs 2              ;stack pointer for this
  117 0011              ltime:  defs 1              ;scheduler timer word
  118 0012              ltdown: defs 2              ;time-down for states
  119 0014              lflag1: defs 1              ;VTN flags (see below)
  120 0015              lflag2: defs 1              ;more VTN flags
  121 0016              ldbtim: defs 1              ;used by DATA for smoo
  122 0017              lcid:   defs 1              ;conversation id
  123 0018              lremb:  defs 1              ;remote vbox if CNVBIT=1
  124 0019              lreml:  defs 1              ;remote link if CNVBIT=
  125 001A              lwatch: defs 1              ;non-zero, watch should
  126 001B              lout:   defs 1              ;number of chars outstan
  127 001C              lin:    defs 1              ;number of chars instan
  128 001D              lrut:   defs nbox           ;routing table for this
  129 002D              llyank: defs 2              ;who called yank last.

  131 002F              ixoffs: equ $               ;size of IX offset area.
  132 007F              lstack: equu lsize-1        ;!!must change IYLIN rou
  133 002F
```

```
;Skeletal buffer/packet format

;Other than line framing and buffer headers, each of the
;VTN packet types are formatted as follows:

; ROUTING PACKETS
;
;          7   6   5   4   3   2   1   0
;        +---+---+---+---+---+---+---+---+
;plen    |          packet length        |
;        +---+---+---+---+---+---+---+---+
;ptob    | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
;        +---+---+---+---+---+---+---+---+
;prup0   |    routing cost to node 0     |
;         . . . . . . . . . . . . . . . .
;        |    routing cost to high node  |
;        +---+---+---+---+---+---+---+---+
;prubp   |      originating V80X         |
;        +---+---+---+---+---+---+---+---+
;prubl   |      originating VLINK        |
;        +---+---+---+---+---+---+---+---+


; RCALL / ACK
;
;          7   6   5   4   3   2   1   0
;        +---+---+---+---+---+---+---+---+
;plen    |          packet length        |
;        +---+---+---+---+---+---+---+---+
;ptob    | 1 |    destination V80X       |
;        +---+---+---+---+---+---+---+---+
;ptol    | A | L |  destination link     | A=1:ack, L=1:
;        +---+---+---+---+---+---+---+---+
;pfrb    |           source V80X         |
;        +---+---+---+---+---+---+---+---+
;pfrl    |           source link         |
;        +---+---+---+---+---+---+---+---+
;paddrl  |      remote subr addr low     |
;        +---+---+---+---+---+---+---+---+
;paddrh  |      remote subr addr high    |
;        +---+---+---+---+---+---+---+---+
;prega   |  copy of call/ret A register  |
;         . . . . . . . . . . . . . . . .
;pregl   |  copy of call/ret L register  |
;        +---+---+---+---+---+---+---+---+
;        |          optional long        |
;         . . . . . . . . . . . . . . . .
;        |           RCALL data          |
;        +---+---+---+---+---+---+---+---+
```

```
; DATA
;
;          7   6   5   4   3   2   1   0
;        +---+---+---+---+---+---+---+---+
;plen    |           packet length       |
;        +---+---+---+---+---+---+---+---+
;ptob    | 0 |     destination V80X      |
;        +---+---+---+---+---+---+---+---+
;ptol    | 0 |     destination link      |
;        +---+---+---+---+---+---+---+---+
;pcid    |       conversation identifier |
;        +---+---+---+---+---+---+---+---+
;ptx1ch  |       first data character    |
;         . . . . . . . . . . . . . . . .
;        |       last data character     |
;        +---+---+---+---+---+---+---+---+

; RELEASE
;
;          7   6   5   4   3   2   1   0
;        +---+---+---+---+---+---+---+---+
;plen    |           packet length       |
;        +---+---+---+---+---+---+---+---+
;ptob    | 0 |     destination V80X      |
;        +---+---+---+---+---+---+---+---+
;ptol    | 1 |     destination link      |
;        +---+---+---+---+---+---+---+---+
;pcid    |       conversation identifier |
;        +---+---+---+---+---+---+---+---+
;pminus  |    amount to decrease LOUT    |
;        +---+---+---+---+---+---+---+---+
```

215 002F

```
216 0000                        org 0

218 0000            pforl:  defs 1              ;forward pointer
219 0001            pforh:  defs 1
220 0002            pbakl:  defs 1              ;backwards pointer
221 0003            pbakh:  defs 1
222 0004            plink:  defs 1              ;link associated with th
223 0005            plen:   defs 1              ;packet's length (numbe
224 0006            ptob:   defs 1              ;final destination vbox
225 0007            prup0:  defs nbox           ;if RUP, cost to node 0
226 0017            prupb:  defs 1              ;node and
227 0018            prupl:  defs 1              ; line for V-Link.
228 0007                    org prup0
229 0007            ptol:   defs 1

231 0008            pcid:                       ;conversation ID for TEX
232 0008            pfrb:   defs 1              ; ... or, from box if SU

234 0009            ptx1ch:                     ;first character for TEX
235 0009            pminus:                     ; ... or, amount to dec
236 0009            pfrl:   defs 1              ; ... or, from link if S

238 0004            prllen: equ 4               ;number of bytes for a
239 0003            ptxovl: equu 3              ;number of overhead byte

241 000A            paddrl: defs 1              ;lo addr for SUBR/ACK
242 000B            paddrh: defs 1              ;hi addr
243 000C            prega:  defs 1              ;registers for SUBR/ACK
244 000D            pregf:  defs 1
245 000E            pregb:  defs 1
246 000F            pregc:  defs 1
247 0010            pregd:  defs 1
248 0011            prege:  defs 1
249 0012            pregh:  defs 1
250 0013            pregl:  defs 1
251 0014            plroff: equ $               ;offset to first long RC
252 000F            prclen: equ $-plen          ;length of an RCALL/ACK
253 0084            bsize:  equ pmax+plen       ;size of a buffer (not
254 0078            ptxnch: equ bsize-ptx1ch    ;number of user text cha
255 0014
```

```
                        ;Flags in LFLAG1

258 0007                cnvbit: equ 7           ;set if in connection
259 0080                cnvmsk: equ 128

261 0006                cmdbit: equ 6           ;set if command decoding
262 0040                cmdmsk: equ 64

264 0005                lrcbit: equ 5           ;long RCALL
265 0020                lrcmsk: equ 32

267 0004                ackbit: equ 4           ;set if this job is wait
268 0010                ackmsk: equ 16

270 0003                vupbit: equ 3           ;V-link up/down
271 0008                vupmsk: equ 8

273 0002                wwbit:  equ 2           ;set if CNV is blocked f
274 0004                wwmsk:  equ 4

                        ;LFLAG2.  Other line status bits

278 0007                eclbit: equ 7           ;local ECHO bit (see EC
279 0080                eclmsk: equ 128

281 0006                rngbit: equ 6           ;have initiated a ring
282 0040                rngmsk: equ 64

284 0005                hedbit: equ 5           ;header written yet?
285 0020                hedmsk: equ 32

287 0003                gbit:   equ 3           ;carrier cleanly gone
288 0008                gmsk:   equ 8
289 0014
```

```
                        ;LTYPE1.  These flag bits do not change during VTN opera
 292 0006               pbkbit: equ 6                   ;pass BREAK through netw
 293 0040               pbkmsk: equ 64

 295 0005               echbit: equ 5                   ;this line echos
 296 0020               echmsk: equ 32

 298 0004               vbit:   equ 4                   ;set if this is a V-link
 299 0010               vmsk:   equ 16                  ; (HBIT must be zero)

 301 0003               onbit:  equ 3                   ;software allowed to use
 302 0008               onmsk:  equ 8

 304 0002               hbit:   equ 2                   ;set if this is an H-lin
 305 0004               hmsk:   equ 4                   ; (if VBIT=HBIT=0, it's

 307 0001               shbit:  equ 1                   ;set if this line does h
 308 0002               shmsk:  equ 2                   ; with full modem contro

 310 0000               dcebit: equ 0                   ;1=DCE, 0=DTE
 311 0001               dcemsk: equ 1


                        ;LTYPE2 - flow control characteristics

 316 0007               gfxbit: equ 7                   ;generate ^S/^Q
 317 0080               gfxmsk: equ 128

 319 0006               pfxbit: equ 6                   ;process ^S/^Q
 320 0040               pfxmsk: equ 64

 322 0005               autbit: equ 5                   ;autoconnect enabled
 323 0020               autmsk: equ 32
 324 0014
```

```
                        ;Group bits

  327 0007              g7bit:  equ 7           ;group 7 (prived)
  328 0080              g7msk:  equ 128
  329 0006              g6bit:  equ 6
  330 0040              g6msk:  equ 64
  331 0005              g5bit:  equ 5
  332 0020              g5msk:  equ 32
  333 0004              g4bit:  equ 4
  334 0010              g4msk:  equ 16
  335 0003              g3bit:  equ 3
  336 0008              g3msk:  equ 8
  337 0002              g2bit:  equ 2
  338 0004              g2msk:  equ 4
  339 0001              g1bit:  equ 1
  340 0002              g1msk:  equ 2
  341 0000              g0bit:  equ 0
  342 0001              g0msk:  equ 1
  343 0014
```

```
                        ;VTN Startup Linkage

346 4000                        org rram

348 4000  C31C53        vtn0:   jp ivtn                 ;VTN start address



                        ;Variable storage for VTN

354 4200                        org vram

356 4200  0F            me:     defb nme                ;my VBOX number
357 4201  0400          vers:   defb minor,major        ;version number (low,hi
358 4203  0D0A0A        name0:  defb ctrlm,ctrlj,ctrlj
359 4206  56544E00      name:   defm 'VTN',0            ;default network name, o
360 420A                        defs 100                ;maximum network name s

362 426E                syssp:  defs 2                  ;system stack pointer
363 4270                cursp:  defs 2                  ;current job stack poi
364 4272                temp1:  defs 2                  ;temps
365 4274                oldclk: defs 1                  ;last MPCO time seen by
366 4275                pplist: defs 4                  ;packets to be process
367 4279                frlist: defs 4                  ;free (available) packet
368 427D                frleng: defs 2                  ;number of packets on FR
369 427F                totbuf: defs 2                  ;total number of buffe
370 4281                vlast:  defs 2                  ;location of first non-e
371 4283                vcheck: defs 2                  ;next mem word to test
372 003C                slowtc: equ 60                  ;sixty fast ticks will
373 4285                                                ;interval timer subrouti
374 4285                                                ;called when timer ticks
375 4285                slow:   defs 1                  ;slow ticker
376 4286                dismax: defs 1                  ;maximum loop time seen.
377 4287                newrut: defs 1                  ;flag for !WATCH to prin
378 4288                xcllok: defs 1                  ;lock on special comma

380 4289                        defs 64
381 42C9                stack:  defs 2                  ;system stack (sched, p

383 42CB                blktab: defs nlink              ;non-zero means output b
384 42EB                routes: defs nbox               ;direction (V-link numb
385 42FB                costs:  defs nbox               ;cost to get there
386 430B                ffs:    defs nbox               ;a table of OFFHs NBOX 1
387 431B                lblk0:  defs lsize              ;line blocks
388 439B                lblk1:  defs lsize
389 441B                lblk2:  defs lsize
390 449B                lblk3:  defs lsize
391 451B                lblk4:  defs lsize*(nlink-4)    ;the rest of them

393 531B                cid:    defs 1                  ;last cid given out (7
394 531C
```

```
                        ;INIT - set up major registers, etc.

 397 531C  3EFF         ivtn:   ld a,mpc99              ;load non-existent-MPC a
 398 531E  D3FF                 out (select),a          ;clear any selected MPC

 400 5320  AF                   xor a                   ;get a zero
 401 5321  ED47                 ld i,a                  ;clear interrupt base
 402 5323  ED4F                 ld r,a                  ;and refresh base
 403 5325  F3                   di                      ;no interrupts, please
 404 5326  328542               ld (slow),a             ;reset slow ticker.
 405 5329  328642               ld (dismax),a           ;clear max dismiss time
 406 532C  31C942               ld sp,stack             ;set up our stack

 408 532F  21034C               ld hl,vdie              ;prepare to clear the m
 409 5332  017A01               ld bc,rstack-vdie       ;load count of area to c
 410 5335  110440               ld de,vdie+1            ;next location to clear.
 411 5338  77                   ld (hl),a               ;load first zero.
 412 5339  EDB0                 ldir                    ;zap VDIE through RSTACK
 413 533B  EEFF                 xor 0FFh                ;clear command lock
 414 533D  328842               ld (xcllok),a
 415 5340  CD8169               call leds               ;and do a LAMP-test duri

                        ;INIT - discover and reload all MPCs.  This code is a m
                        ;machine isn't up yet.  To facilitate processing, we jus

 420 5343  FD211843     impc:   ld iy,lblk0             ;get a dummy line block
 421 5347  AF                   xor a
 422 5348  FD770D       implop: ld (iy+lnum),a          ;store the line number
 423 5348  CD8B69               call iynum              ;get MPC number into A
 424 534E  D3FF                 out (select),a          ;try to select it
 425 5350  3A660C               ld a,(066h)             ;did it select?
 426 5353  FEC3                 cp 0C3h
 427 5355  2018                 jr nz,impnxt            ;nope, try to select nex
 428 5357  3A0400               ld a,(mpunv)            ;load universal file num
 429 535A  FE00                 cp munv                 ;is it correct?
 430 535C  C403FC               call nz,rdie            ; ++ IMPLOP: universal s

                        ;Now beat on this MPC a little by selecting and de-sele

 434 535F  0600                 ld b,0                  ;do it 256 times..
 435 5361  CD9569       impsll: call mpcsel             ;select (and check) it
 436 5364  CDA369               call mpcuns             ;then de-select it
 437 5367  CDAE69               call unschk             ;and check it
 438 536A  10F5                 djnz impsll             ;a lot of times!

 440 536C  CD156B               call mpcnmi             ;NMI it

 442 536F  3A2843       impnxt: ld a,(lblk0+lnum)       ;get line number back
 443 5372  C604                 add a,4                 ;else step to next
 444 5374  FE20                 cp nlink                ;hit max?
 445 5376  20D0                 jr nz,implop
 446 5378
```

```
                        ;INIT - clear and size packet memory

 449 5378  218872       iclear: ld hl,vbuf0             ;get ptr to start of buf
 450 537B  36FF         clrlop: ld (hl),0ffh            ;write OFFH
 451 537D  3600                 ld (hl),0               ;then zero
 452 537F  7E                   ld a,(hl)               ;read it back
 453 5380  B7                   or a
 454 5381  2005                 jr nz,clrend            ;didn't get zero back,
 455 5383  23                   inc hl                  ;step to next
 456 5384  7C                   ld a,h                  ;make sure we didn't wra
 457 5385  B5                   or l
 458 5386  20F3                 jr nz,clrlop            ;around to zero...

 460 5388  228142       clrend: ld (vlast),hl           ;save end of world's ad
 461 538B                                               ;one beyond world's end)

                        ;Now, knowing the end of memory, call MEMCHK at full sp

 465 538B  0602                 ld b,2                  ;times to do this
 466 538D  C5           imemc1: push bc                 ;save counter on top of
 467 538E  210042               ld hl,vram              ;start at VRAM
 468 5391  CD0765       imemc2: call memchk             ;check this word
 469 5394  23                   inc hl
 470 5395  ED488142             ld bc,(vlast)
 471 5399  AF                   xor a                   ;set CY=0
 472 539A  E5                   push hl
 473 539B  ED42                 sbc hl,bc
 474 539D  E1                   pop hl                  ;save HL over SBC in cas
 475 539E  20F1                 jr nz,imemc2            ;if not at end, loop
 476 53A0  C1                   pop bc                  ;else start again
 477 53A1  10EA                 djnz imemc1

                        ;INIT - carve up memory into buffers on the free list

 481 53A3  217942       icarve: ld hl,frlist            ;initialize free list h
 482 53A6  CD3D54               call ihedr
 483 53A9  210000               ld hl,0                 ;clear FRLENG
 484 53AC  227D42               ld (frleng),hl
 485 53AF  217542               ld hl,pplist            ;initialize packet list
 486 53B2  CD3D54               call ihedr

 488 53B5  2A8142               ld hl,(vlast)           ;start at the end of the
 489 53B8  017CFF       crvlop: ld bc,-bsize            ;compute where this one
 490 53BB  09                   add hl,bc               ;hl=hl-bsize..
 491 53BC  E5                   push hl                 ;save copy
 492 53BD  018972               ld bc,vbuf0+1           ;compare with end of VTN
 493 53C0  AF                   xor a                   ;clear carry
 494 53C1  ED42                 sbc hl,bc
 495 53C3  E1                   pop hl
 496 53C4  3815                 jr c,crvend             ;if negative, all done

 498 53C6  E5           crvenq: push hl                 ;put packet pointer in i
 499 53C7  DDE1                 pop ix
 500 53C9  217942               ld hl,frlist            ;get ptr to free list he
 501 53CC  CD5C69               call enq                ;enq this free packet
 502 53CF  2A7D42               ld hl,(frleng)
 503 53D2  23                   inc hl                  ;bump free count
```

```
 504 53D3  227D42            ld (frleng),hl
 505 53D6  DDE5              push ix
 506 53D8  E1                pop hl                  ;copy top address into
 507 53D9  18DD              jr crvlop               ;and try next

 509 53DB  2A7D42    crvend: ld hl,(frleng)          ;get free list size
 510 53DE  227F42            ld (totbuf),hl          ;save it for later check
 511 53E1  210042            ld hl,vram              ;start memory checker at
 512 53E4  228342            ld (vcheck),hl
 513 53E7
```

```
                        ;INIT - set up routing tables

516 53E7  3EFF      irut:   ld a,0ffh               ;get an FF
517 53E9  328742            ld (newrut),a           ;say there's new routing
518 53EC  21F842            ld hl,costs             ;set COSTS to infinite
519 53EF  77                ld (hl),a               ;the first entry..
520 53F0  11FC42            ld de,costs+1
521 53F3  010F00            ld bc,nbox-1            ;it's this big
522 53F6  EDB0              ldir                    ;all of them

524 53F8  210B43            ld hl,ffs               ;set up the FFs constant
525 53FB  110C43            ld de,ffs+1
526 53FE  010F00            ld bc,nbox-1
527 5401  77                ld (hl),a               ;load the first 0FFh
528 5402  EDB0              ldir                    ;propogate them.

                        ;INIT - set up jobs and init MBLKs

532 5404  FD211B43  ijob:   ld iy,lblk0             ;start with line zero
533 5408  0620              ld b,nlink              ;for NLINK of them
534 540A  3E20      ijobl:  ld a,nlink              ;compute link number
535 540C  90                sub b
536 540D  FD770D            ld (iy+lnum),a
537 5410  C5                push bc
538 5411  CD4954            call zaplin             ;clear out the line bloc
539 5414  AF                xor a
540 5415  FD7718            ld (iy+lremb),a         ;clear out the variables
541 5418  FD7719            ld (iy+lreml),a
542 541B  FD771A            ld (iy+lwatch),a
543 541E  C1                pop bc
544 541F  114056            ld de,dead              ;start off in DEAD
545 5422  CDA164            call yankhm
546 5425  CDAE69            call unschk             ;[aej] check
547 5428  118000    ijobml: ld de,lsize
548 542B  FD19              add iy,de               ;step to next
549 542D  10D8              djnz ijobl              ;if there is one

                        ;INIT - end of initializtion

553 542F  CD3355    iend:   call gtime              ;get current time.
554 5432  327442            ld (oldclk),a           ;store initial value.

556 5435  3E82              ld a,vtnup+pplmty       ;VTN is up, proccessing
557 5437  CD8169            call leds               ;update display
558 543A  C37854            jp nxtjob               ;start scheduling
559 543D
```

```
                        ;Initialize queue header to point to itself.
                        ;Call with header's address in HL.

563 543D  44            ihedr:  ld b,h                  ;copy to safer place
564 543E  4D                    ld c,l
565 543F  71                    ld (hl),c               ;set PFORL
566 5440  23                    inc hl
567 5441  70                    ld (hl),b               ;PFORH
568 5442  23                    inc hl
569 5443  71                    ld (hl),c               ;PBAKL
570 5444  23                    inc hl
571 5445  70                    ld (hl),b               ;PBAKH
572 5446  60                    ld h,b
573 5447  69                    ld l,c                  ;fix HL
574 5448  C9                    ret


                        ;Subroutine to clear up a line block to its initial stat
                        ;do disconnections, though), then reset the line in the

580 5449  AF            zaplin: xor a                   ;clear random parameters
581 544A  FD7714                ld (iy+lflag1),a
582 544D  FD7715                ld (iy+lflag2),a
583 5450  FD7711                ld (iy+ltime),a
584 5453  FD7717                ld (iy+lcid),a
585 5456  FD7718                ld (iy+lout),a
586 5459  FD771C                ld (iy+lin),a

588 545C  FDCB006E              bit echbit,(iy+ltype1)
589 5460  2804                  jr z,zapech
590 5462  FDCB15FE              set eclbit,(iy+lflag2)

592 5466  FDCB005E    zapech:   bit onbit,(iy+ltype1)   ;is line active?
593 546A  C43C68                call nz,mpczpl          ;clear the line

595 546D  210843                ld hl,ffs               ;make routing infinite
596 5470  FDCB005E              bit onbit,(iy+ltype1)   ;is line active?
597 5474  C44865                call nz,rup
598 5477  C9                    ret
599 5478
```

```
                        ;This is the top of the scheduler code.  There are two s
                        ;NXTJOB routine which finds the next runnable job and t

                        ;NXTJOB computes the DELTA time since it last ran by get
                        ;from MPC0.  Then all jobs' LTIMEs are decremented by D
                        ;LTIME has counted down it is run.  After line blocks, P

607 5478  CD3355        nxtjob: call gtime              ;get first MPC's ticke
608 547B  217442                ld hl,oldclk            ;get pointer to last tim
609 547E  46                    ld b,(hl)               ;read it (max=255, min=0
610 547F  77                    ld (hl),a               ;store new current time
611 5480  90                    sub b                   ;compute current-last
612 5481  F28654                jp p,nxtj0              ;skip on if this is posi
613 5484  ED44                  neg                     ;adjust difference if
614 5486  4F            nxtj0:  ld c,a                  ;move delta to useful pl
615 5487  218642                ld hl,dismax            ;load previous max dismi
616 548A  BE                    cp (hl)                 ;is this difference gr
617 548B  3801                  jr c,dism0              ;no, skip on...
618 548D  77                    ld (hl),a               ;replace time value.

620 548E  3A8542        dism0:  ld a,(slow)             ;load slow ticker count.
621 5491  91                    sub c                   ;compute difference.
622 5492  3009                  jr nc,nxtsl0            ;no carry, just store i
623 5494  C63C                  add a,slowtc            ;adjust time count.
624 5496  47                    ld b,a
625 5497  C5                    push bc
626 5498  CDFB54                call slowsb             ;call slow ticker subrou
627 549B  C1                    pop bc
628 549C  78                    ld a,b
629 549D  328542        nxtsl0: ld (slow),a             ;restore count.

631 54A0  FD211B43              ld iy,lblk0             ;start at line zero
632 54A4  0620                  ld b,nlink              ; for all lines.

634 54A6  FD7E11        nxtlop: ld a,(iy+ltime)         ;get its blocking time
635 54A9  91                    sub c                   ;reduce by delta
636 54AA  3871                  jr c,runit              ;if went negative
637 54AC  286F                  jr z,runit              ;or equal, run it
638 54AE  FD7711                ld (iy+ltime),a         ;update time

640 54B1  11B000        nxtret: ld de,lsize             ; (enter here after run
641 54B4  FD19                  add iy,de               ;step to next
642 54B6  10EE                  djnz nxtlop             ;go through them all
643 54B8
```

```
                        ;Here when they've all been run (as needed), call PPL an

646 54B8  CD2166                call ppl
647 54BB  CDAE69                call unschk             ;make sure it's un-selec
648 54BE  CD8866                call rel
649 54C1  CDAE69                call unschk             ;make sure it's un-selec
650 54C4  2A7D42                ld hl,(frleng)          ;get the free length
651 54C7  ED4B7F42              ld bc,(totbuf)          ;and the total number o
652 54CB  AF                    xor a                   ;clear carry
653 54CC  ED42                  sbc hl,bc               ;compute difference
654 54CE  7C                    ld a,h
655 54CF  B5                    or l
656 54D0  3A8241                ld a,(patrn)            ;get lights pattern
657 54D3  2804                  jr z,empty
658 54D5  E6FD                  and 0FFh-pplmty         ;clear empty bit
659 54D7  1802                  jr ppllit
660 54D9  F602          empty:  or pplmty               ;set empty
661 54DB  CD8169        ppllit: call leds

663 54DE  2A8342                ld hl,(vcheck)          ;get ptr to next word t
664 54E1  CD0765                call memchk             ;check it out
665 54E4  23                    inc hl                  ;at end?
666 54E5  228342                ld (vcheck),hl          ;assume not
667 54E8  ED5B8142              ld de,(vlast)
668 54EC  AF                    xor a                   ;set CY=0
669 54ED  ED52                  sbc hl,de
670 54EF  C27854                jp nz,nxtjob            ;if not at end, loop
671 54F2  210042                ld hl,vram
672 54F5  228342                ld (vcheck),hl
673 54F8  C37854                jp nxtjob               ;else re-set to top of t

                        ;Count each line's LTDOWN to zero.

677 54FB  FD211843      slowsb: ld iy,lblk0             ;start at block 0
678 54FF  0620                  ld b,nlink              ; for all lines.

680 5501  FD6E12        slolop: ld l,(iy+ltdown)        ;get double
681 5504  FD6613                ld h,(iy+ltdown+1)      ; precision timer.
682 5507  7D                    ld a,l                  ;is it zero?
683 5508  B4                    or h
684 5509  280A                  jr z,slonxt             ;is it already zero - i
685 550B  11FFFF                ld de,-1                ;decrement by one.
686 550E  19                    add hl,de
687 550F  FD7512                ld (iy+ltdown),l        ;restore timer value.
688 5512  FD7413                ld (iy+ltdown+1),h

690 5515  118000        slonxt: ld de,lsize             ;load size of line bloc
691 5518  FD19                  add iy,de               ; and step to next line
692 551A  10E5                  djnz slolop             ; if there is one.
693 551C  C9                    ret
694 551D
```

```
                        ;Here when job in IY is runnable.  Save main line's BC.
                        ;It is the line job's responsibility to respect IY.

 698 551D  C5           runit:  push bc
 699 551E  ED736E42             ld (syssp),sp           ;save our stack pointer
 700 5522  FD661C               ld h,(iy+1sp+1)         ;set up HL with his SP
 701 5525  FD6E0F               ld l,(iy+1sp+0)
 702 5528  F9                   ld sp,hl
 703 5529  227042               ld (cursp),hl           ;remember this job for d

 705 552C  DDE1                 pop ix                  ;restore his context
 706 552E  E1                   pop hl
 707 552F  D1                   pop de
 708 5530  C1                   pop bc
 709 5531  F1           runpaf: pop af
 710 5532  C9                   ret                     ;return to him

 712 5533  3EF0         gtime:  ld a,mpc0               ;get lowest numbered MPC
 713 5535  CD9869               call mpcsla             ;select (and check) MPC.
 714 5538  3A0D20               ld a,(mvtime)           ;get its clock word
 715 553B  C3A369               jp mpcuns               ;release MPC and return.
 716 553E
```

```
                        ;Various DB routines

 719 553E  F5           db:     push af
 720 553F  AF                   xor a                   ;no time at all...
 721 5540  180D                 jr itran

 723 5542  F5           db1:    push af
 724 5543  3E78                 ld a,120                ;120 ticks/second
 725 5545  1808                 jr itran

 727 5547  F5           dbr:    push af
 728 5548  3E04                 ld a,120/30             ;30 cps typing rate
 729 554A  1803                 jr itran

 731 554C  F5           db01:   push af
 732 554D  3E0C                 ld a,12                 ;.1 seconds

                        ;Here from DB routines after setting up LTIME

 736 554F  FD7711       itran:  ld (iy+ltime),a         ;store debrk time.
 737 5552  C5                   push bc
 738 5553  D5                   push de
 739 5554  E5                   push hl
 740 5555  DDE5                 push ix

 742 5557  210000               ld hl,0
 743 555A  39                   add hl,sp               ;load copy of stack addr
 744 555B  FD741C               ld (iy+lsp+1),h
 745 555E  FD750F               ld (iy+lsp+0),l         ;update LSP
 746 5561  ED7B6E42             ld sp,(syssp)           ;restore system stack
 747 5565  CDAE69               call unschk             ;make sure MPC is de-sel

 749 5568  C1                   pop bc
 750 5569  C3B154              jp nxtret               ;return into nextjob loo
 751 556C
```

```
                        ;This is the packet reader, PKTIN. It has a few importan

                        ;1. read in new packets from the V-link and queue them f
                        ;   PLINK to be the value of the link via which they arr

                        ;2. processing incoming routing updates.

                        ;3. scheduling routing updates for output every RUPTIM

                        ;4. watching for missing RUPs from a V-link and updating
                        ;       (RUT) to mark the V-link dead.

                        ;The time cycle of the RUP interchanges is started at PK
                        ;time-to-transmission of the RUP message to one half of
                        ;message is transmitted, the time to transmission of the
                        ;one full RUPTIM.  Should that time expire, the counter
                        ;When the counter becomes zero, the link is declared do
                        ;that the neighbor will respond "soon" after the one hal
                        ;the reinitialization of the loop at PKTINI before a ful

772 556C 110200         pktini: ld de,rupnum            ;load the counter and
773 556F D5                     push de                 ; save it on the stack.
774 5570 010100                 ld bc,ruptim/2          ;use 0.5 interval until

776 5573 CDFE64         pktins: call settd              ;set the time-down

778 5576 CD3E55         pktin:  call db                 ;wait a little while
779 5579 CDEE6A                 call mpckbc             ;any data?
780 557C 2850                   jr z,pkttim             ;no - skip input and go

782 557E CDDA68                 call getbuf             ;get a buffer to hold in
783 5581 DD7704                 ld (ix+plink),a         ;save supposed length

785 5584 DDE5                   push ix                 ;compute read in address
786 5586 E1                     pop hl
787 5587 010500                 ld bc,plen              ;where to start input
788 558A 09                     add hl,bc

790 558B 3E7F                   ld a,pmax               ;get max size
791 558D 0602                   ld b,mrp                ;get right command
792 558F CD0C6A                 call mpcin              ;read it
793 5592 CC03FC                 call z,rdie             ; ++ PKTIN: MPCIN failed
794 5595 FD7E0D                 ld a,(iy+lnum)          ;copy the line number
795 5598 DD7704                 ld (ix+plink),a
796 559B DD7E06                 ld a,(ix+ptob)          ;is this a routing messa
797 559E 3C                     inc a                   ;check it by incrementin
798 559F 2C2D                   jr nz,pkttim            ;if not, go check timer
799 55A1
```

```
                        ;Here when there's a routing update from the line. Re-se
                        ;time-down to RUPNUM*RUPTIM units and process the rout

 803 55A1  CDC655       pktrup: call prup0x             ;get index to box zero i
 804 55A4  CD4B65               call rup                ;do the update
 805 55A7  DD7E17               ld a,(ix+prupb)         ;load other node number
 806 55AA  FD7718               ld (iy+lremb),a         ; and store as remote bo
 807 55AD  DD7E18               ld a,(ix+prupl)         ;do same with link numb
 808 55B0  FD7719               ld (iy+lreml),a
 809 55B3  FDCB145E             bit vupbit,(iy+lflag1)  ;state already up?
 810 55B7  2007                 jr nz,pktaru
 811 55B9  FD341A               inc (iy+lwatch)         ;no, mark for WATCH
 812 55BC  FDCB14DE             set vupbit,(iy+lflag1)  ;set V-link verified bit
 813 55C0  CD0269       pktaru: call givbuf             ;we're done with the bu
 814 55C3  D1                   pop de                  ;get number of RUPs left
 815 55C4  18A6                 jr pktini               ;reinitialize

                        ;PRUP0X computes index to information about box zero in
                        ;Destroys BC, returns index in HL.

 820 55C6  DDE5         prup0x: push ix                 ;compute where row is in
 821 55C8  E1                   pop hl                  ;into HL
 822 55C9  010700               ld bc,prup0             ;add offset of first bo
 823 55CC  09                   add hl,bc               ; packet base -> result
 824 55CD  C9                   ret

                        ;Here when we've exhausted all input, check to see if TD
                        ;call RUP with all infinite cost and MPCZPL the line to

 829 55CE  CDF764       pkttim: call tdownp             ;did we time down?
 830 55D1  20A3                 jr nz,pktin             ;nope, dismiss and check
 831 55D3  C1                   pop bc                  ;time to declare line d
 832 55D4  79                   ld a,c                  ;had top of stack gone t
 833 55D5  B0                   or b                    ;set CCR
 834 55D6  200F                 jr nz,pktsup            ;still more ticks left,
 835 55D8  FDCB145E             bit vupbit,(iy+lflag1)  ;already down?
 836 55DC  2803                 jr z,pktded             ;yes, skip INC of LWATCH
 837 55DE  FD341A               inc (iy+lwatch)         ;for WATCH
 838 55E1  114056       pktded: ld de,dead              ;prepare to kill the lin
 839 55E4  CD8D64               call yankme             ;if top of stack went to

 841 55E7                       ;fall into PKTSUP
 842 55E7
```

```
843 55E7                        ;from previous page

                        ;Here when the line has timed down another RUPTIM unit (
                        ;the line).  Send a routing update and restart the timer
                        ;are always placed at the front of the PPLIST so they g
                        ;packets.

850 55E7  0B            pktsup: dec bc                  ;tick down one and plac
851 55E8  C5                    push bc                 ; decremented count onto

853 55E9  CDDA68                call getbuf             ;get a buffer
854 55EC  CD1369                call deq                ;move it to front of PPL
855 55EF  2A7542                ld hl,(pplist)          ;before first entry
856 55F2  CD5C69                call enq

858 55F5  DD3606FF              ld (ix+ptob),0FFh       ;set up the routing code
859 55F9  DD360513              ld (ix+plen),nbox+1+2   ;size of the update (+2
860 55FD                                                ; remote node and link n
861 55FD  FD7E0D                ld a,(iy+lnum)          ;copy our number into PL
862 5600  DD7704                ld (ix+plink),a         ;(PLINK doesn't get tra
863 5603  CDC655                call prup0x             ;compute base of array,
864 5606  E5                    push hl                 ;save address of first b
865 5607  EB                    ex de,hl                ;put it into DE
866 5608  21FB42                ld hl,costs             ;get costs pointer
867 560B  011000                ld bc,nbox              ;size of transfer
868 560E  EDB0                  ldir                    ;copy min delay table i
869 5610  D1                    pop de                  ;load address of zeroth
870 5611  21EB42                ld hl,routes            ;load address of route t
871 5614  0600                  ld b,0                  ;do for all boxes.
872 5616  DD7E04        pktlop: ld a,(ix+plink)         ;is the link in question
873 5619  BE                    cp (hl)                 ; route to the target bo
874 561A  200A                  jr nz,pktnxl            ;nope, skip to next.
875 561C  3A0042                ld a,(me)               ;load my box number to c
876 561F  B8                    cp b                    ; is this me.  If so, RO
877 5620  2804                  jr z,pktnxl             ; and I shouldn't repor
878 5622  EB                    ex de,hl                ;exchange pointer of rou
879 5623  36FF                  ld (hl),0ffh            ;make this cost infinite
880 5625                                                ; master of the situati
881 5625  EB                    ex de,hl                ;restore registers to no
882 5626  23            pktnxl: inc hl                  ;increment to next route
883 5627  13                    inc de                  ;increment to next pack
884 5628  04                    inc b                   ;point to the box in que
885 5629  78                    ld a,b                  ;check to see if box to
886 562A  FE11                  cp nbox+1               ; checked is within ran
887 562C  20E8                  jr nz,pktlop            ;repeat if not done.

889 562E  010200                ld bc,ruptim            ;load full wait time.
890 5631  3A0042                ld a,(me)
891 5634  DD7717                ld (ix+prupb),a         ;store my box number.
892 5637  FD7E0D                ld a,(iy+lnum)
893 563A  DD7718                ld (ix+prupl),a
894 563D  C37355                jp pktins               ;send it off, and sleep
895 5640
```

```
                        ;This is the DEAD state.  We kill the interface, etc. an
                        ;(if V-link), CMD (if T-link with a character) or eat

 899 5640  CD1E64       dead:   call discon             ;get rid of any connecti
 900 5643  CD1264               call givlok             ;and give back any comm
 901 5646  CD4255       deadwt: call db1                ;wait a while.
 902 5649  FDCB0C5E             bit onbit,(iy+ltype1)   ;test for line active.
 903 564D  28F7                 jr z,deadwt             ;if line is not on, do

 905 564F  CD4954               call zaplin             ;return everything else
 906 5652  FDCB0066             bit vbit,(iy+ltype1)    ;a V-link?
 907 5656  C26C55               jp nz,pktini            ;yes, start up packet I/

                        ;&& because of the fact that it takes the -20 a long ti
                        ;&& sit here waiting for 1 minute with the line offline
                        ;&& getting this line in a connection - only to have it

 913 5659  FDCB155E             bit gbit,(iy+lflag2)    ;did host's carrier drop
 914 565D  FDCB159E             res gbit,(iy+lflag2)    ;turn off "gone" bit
 915 5661  2018                 jr nz,deadw             ;yes, all is well else

 917 5663  FD7E00               ld a,(iy+ltype1)        ;a DCE with handshaking?
 918 5666  E603                 and shmsk+dcemsk
 919 5668  FE03                 cp shmsk+dcemsk
 920 566A  200F                 jr nz,deadw             ;no, just skip along

 922 566C  FDCB009E             res onbit,(iy+ltype1)   ;take the DCE offline fo
 923 5670  063C                 ld b,60                 ;one minute for DCE
 924 5672  CD4255       deadxw: call db1
 925 5675  10FB                 djnz deadxw             ;waste some time
 926 5677  FDCB00DE             set onbit,(iy+ltype1)   ;bring the line back fro
                        ;&& end of kluge
 928 567B
```

```
 929 567B  CD4C55      deadw:  call db01
 930 567E  CD306B              call active             ;is there any activity
 931 5681  28F8                jr z,deadw              ;no, don't waste MPC ban

 933 5683  CDCE6A              call mpcsta             ;get MPC status
 934 5686  CB7F                bit gonbit,a            ;has carrier gone?
 935 5688  20B6                jr nz,dead              ;yup, zap line again.

 937 568A  CDEE6A              call mpckbc             ;wait for a character
 938 568D  28EC                jr z,deadw

 940 568F  FDCB0056            bit hbit,(iy+ltype1)
 941 5693  CA1859              jp z,cmd                ;if TLINK, let it into c
 942 5696  CDC15F              call clrbfi             ;if HLINK, eat up the c
 943 5699  18E0                jr deadw

                       ;Enter here from various places with the address of a s
                       ;then CRLF.  We type the string and call CLRBFI to gobbl
                       ;up for output to finish then and enter the normal part
                       ;We only output the message if the line is an ON T-link

 950 569B  FDCB0C66    deado:  bit vbit,(iy+ltype1)    ;a V-link?
 951 569F  209F                jr nz,dead              ;yeah - skip it
 952 56A1  FD7E00              ld a,(iy+ltype1)
 953 56A4  E60C                and onmsk+hmsk
 954 56A6  FE08                cp onmsk                ;an H-link (or OFF)?
 955 56A8  2006                jr nz,deads
 956 56AA  CD9362              call strout             ;all's well, output the
 957 56AD  CD9062              call crlf               ; plus a crlf.
 958 56B0  CDC15F      deads:  call clrbfi             ;clear input buffer.
 959 56B3  063C                ld b,60                 ;give output 1 minute (m
 960 56B5  CD036B      deadow: call mpcptl             ;how much is left?
 961 56B8  EEFF                xor 0FFh                ;actually, how much used
 962 56BA  CD4255              call db1                ;wait a second in any ca
 963 56BD  CA4056              jp z,dead               ;if none left after 1 s
 964 56C0  10F3                djnz deadow             ;else wait up to 1 minut
 965 56C2  C34056              jp dead                 ;then give up anyway
 966 56C5
```

```
;VTN DATA SMOOTHING ALGORITHM

;The DATA state uses a power-of-two smoothing algorithm
;the network overhead down to a minimum.  Every time the
;keyboard has been typing "full-out", we double the de-
;maximum of DBMAX).  Every time the keyboard has not, we
;minimum of DBMIN).  If the keyboard has no characters a
;de-break time back to DBMIN. In this way, DATA tends
;full packets.  DBMIN=dbr (1/30 second == 30 CPS), DBMAX
;following table illustrates the number of characters pe
;baud rates:
;
;                1/30    2/30    4/30
;               ----------------------
;          300 !    1       2       4
;         1200 !    4       8      16
;         4800 !   16      32      64
;         9600 !   32      64    *128
;
;"*" denotes the case where the number of characters is
;maximum packet size. One must be careful when choosing
;baud line, 4/30 of a second yields 128 characters. If o
;(8/30 sec), that would yield 256 characters which woul
;flow control logic to step in and XOFF the line after 2
;current value of DBMAX is chosen to avoid flow control
;19,200 baud lines, will, of course, be XOFF'ed.  There
;characters per TEXT packet (start char, length, TO8, TO
;The following table shows the overhead incurred for the
;table. Overhead is defined as 6/(6+text chars).
;
;                1/30    2/30    4/30
;               ----------------------
;          300 !   85      75      60
;         1200 !   60      42      27
;         4800 !   27      15       8
;         9600 !   15       8      *4
```

```
                        ;VTN END-END ALGORITHM

                        ;In a VTN virtual circuit, the sender always sends few e
                        ;such that the receiver can immediately place them in t
                        ;for the destination link.  That is, the sender never ge
                        ;characters ahead of the receiver (see TEXT:'s handling
                        ;packets).

                        ;To enforce this, the sender maintains a counter of the
                        ;outstanding in LOUT.  Every time it sends "s" characte
                        ;by "s".  When LOUT=255, the sender must block until roo

                        ;The receiver maintains a counter equalling the number
                        ;"thinks" the sender has used.  This is known as the "in
                        ;LIN.  In theory, LIN tracks LOUT albeit with some time
                        ;two counters are separated some arbitrary delay, the r
                        ;will watch LIN - if it grows "too large", it will send
                        ;back to the sender.  The RELEASE carries a number which
                        ;LOUT thus making fewer characters outstanding.  In thi
                        ;will generally RELEASE LOUT before it becomes 255 and b

                        ;Since VTN does not provide reliable transmission, it i
                        ;guarentee that the TEXT and RELEASE packets make it thr
                        ;This can lead to skews between LIN and LOUT.  A subrout
                        ;use RCALLs to synchronize the two counts.  The sender
                        ;the count reaches 255 or every 3 seconds when there is
                        ;are characters outstanding.  This procedure, although i
                        ;help recover after sync loss and will always maintain
                        ;window.
1034 56C5
```

```
                        ;DATA state. Enter at DATA0 from YANKs. Note that DBMIN
                        ;units of DBR. LDBTIM counts from DBMIN to DBMAX. The
                        ;2**(LDBTIM-1).

1039 0001               dbmin:  equ 1                   ;DBR * 2**0 = 1*DBR (if
1040 0003               dbmax:  equ 3                   ;DBR * 2**2 = 4*DBR

1042 56C5 FD361601      data0:  ld (iy+ldbtim),dbmin    ;assume DBMIN time firs
1043 56C9 FD361C00              ld (iy+lin),0           ;assume nothing instandi
1044 56CD FD361BFF              ld (iy+lout),255        ;assume everything outst
1045 56D1 CD8558        data0a: call sync               ;start the connection o
1046 56D4 FD7E1B                ld a,(iy+lout)          ;room?
1047 56D7 3C                    inc a
1048 56D8 28F7                  jr z,data0a             ;no - wait for it

                        ;This is the top of the main DATA loop.  Wait for charac
                        ;When they do, try to allocate some window space for th

1053 56DA CDE358        data1:  call remchk             ;is there a remote to se
1054 56DD CDEE6A                call mpckbc             ;any characters?
1055 56E0 CA8557                jp z,wait               ;if not, just wait for s

                        ;Here when there are A characters to be sent.  Make sur
                        ;space to send them in.  If the send window is full, cal
                        ;it open after waiting 1 second.

1061 56E3 57                    ld d,a                  ;save character count
1062 56E4 FD7E1B                ld a,(iy+lout)          ;are there 255 already o
1063 56E7 3C                    inc a
1064 56E8 2C16                  jr nz,data2             ;no, go directly to DATA
1065 56EA FDCB14D6              set wwbit,(iy+lflag1)   ;set the window wait fla
1066 56EE CD4255                call db1                ;wait up to 1 second
1067 56F1 FDCB1496              res wwbit,(iy+lflag1)   ;not waiting anymore
1068 56F5 FD7E1B                ld a,(iy+lout)          ;are there still 255 out
1069 56F8 3C                    inc a
1070 56F9 2C05                  jr nz,data2             ;no, go directly to DATA
1071 56FB CD8558                call sync               ;yes, get the ends synch
1072 56FE 18DA                  jr data1
1073 5700
```

```
                        ;Armed with number of characters in KBD in D, compute ne
                        ;fetching the number of characters we would have gotten
                        ;multiplying by LDBTIM and comparing that with the numbe

1078 5700  D5           data2:  push de                 ;save KBC (in D)
1079 5701  CDDE6A               call mpcrat             ;get the MRATE word into
1080 5704  FEFF                 cp bhunt                ;is it hunting?
1081 5706  2002                 jr nz,data2a
1082 5708  3E0E                 ld a,b9600              ;yes - assume 9600 baud
1083 570A  1600         data2a: ld d,0
1084 570C  5F                   ld e,a                  ;find NTAB(rate)
1085 570D  210358               ld hl,ntab
1086 5710  19                   add hl,de
1087 5711  7E                   ld a,(hl)               ;now A=number of chars
1088 5712  FD4616               ld b,(iy+ldbtim)        ;get DBTIME (1,2,3..)

1090 5715  1002         dataml: djnz datam2             ;multiply by 2**(DBTIME
1091 5717  1805                 jr datam9
1092 5719  87           datam2: add a,a
1093 571A  30F9                 jr nc,dataml            ;loop until goes inifi
1094 571C  3EFF                 ld a,0FFh               ;FF chars if overflow
1095 571E  47           datam9: ld b,a                  ;save number of characte
1096 571F  F1                   pop af                  ;get back number we go
1097 5720  4F                   ld c,a                  ;save it
1098 5721  90                   sub b                   ;compute difference
1099 5722  3C                   inc a                   ;if one short, make 0
1100 5723  1E01                 ld e,1                  ;assume we're going to i
1101 5725  F22A57               jp p,datacd             ;if positive, we assumed
1102 5728  1EFF                 ld e,-1                 ;else, decrement it
1103 572A  FD7E16       datacd: ld a,(iy+ldbtim)        ;get current DB time
1104 572D  83                   add a,e                 ;update it
1105 572E  FE01                 cp dbmin                ;is it too small now?
1106 5730  3004                 jr nc,datac2
1107 5732  3E01                 ld a,dbmin              ;yes, make it DBMIN
1108 5734  1806                 jr datac9
1109 5736  FE04         datac2: cp dbmax+1              ;too big?
1110 5738  3802                 jr c,datac9
1111 573A  3E03                 ld a,dbmax              ;yes, make it DBMAX
1112 573C  FD7716       datac9: ld (iy+ldbtim),a
1113 573F  79                   ld a,c                  ;restore keyboard counte

1115 5740                       ;fall to next page
1116 5740
```

```
1117 5740                      ;from previous page

                       ;Here having # of characters we want to output (MIN[wind
                       ;See if it will fit in a packet - truncating as necessar

1122 5740  FE7B                cp ptxnch               ;will it fit?
1123 5742  3802                jr c,checkw             ;if so, just send it
1124 5744  3E7A                ld a,ptxnch-1           ;if not, truncate to on

                       ;Increase the number of characters outstanding.

1128 5746  FDE5        checkw: push iy
1129 5748  E1                  pop hl
1130 5749  011B00              ld bc,lout              ;get pointer to LOUT
1131 574C  09                  add hl,bc
1132 574D  47                  ld b,a                  ;save original desired o
1133 574E  86                  add a,(hl)              ;increase number outsta
1134 574F  3005                jr nc,check2            ;if didn't go beyond 255
1135 5751  90                  sub b                   ;same as LD A,(HL) - rea
1136 5752  2F                  cpl                     ;compute 255-out = left
1137 5753  47                  ld b,a                  ;change amount we need
1138 5754  3EFF                ld a,255                ;and update LOUT
1139 5756  77          check2: ld (hl),a               ;update LOUT, fall into
1140 5757  C5          sendit: push bc                 ;save length (in B)
1141 5758  CDDA68              call getbuf             ;get a buffer for the te
1142 575B  F1                  pop af                  ;get length back into A
1143 575C  5F                  ld e,a                  ;save a copy for a momen
1144 575D  C603                add a,ptxovl            ;include packet overhead
1145 575F  DD7705              ld (ix+plen),a
1146 5762  7B                  ld a,e                  ;get un-bastardized copy

1148 5763  DDE5                push ix                 ;compute transfer param
1149 5765  E1                  pop hl
1150 5766  010900              ld bc,ptx1ch            ;form address of first c
1151 5769  09                  add hl,bc
1152 576A  0601                ld b,mrt                ;read text from MPC into
1153 576C  CD0C6A              call mpcin              ;get them
1154 576F  BB                  cp e                    ;make sure we got what
1155 5770  C403FC              call nz,rdie            ; ++ DATA: MPCIN gave wr

1157 5773  FD7E17              ld a,(iy+lcid)          ;fill in the rest of th
1158 5776  DD7708              ld (ix+pcid),a
1159 5779  FD7E18              ld a,(iy+lremb)
1160 577C  DD7706              ld (ix+ptob),a
1161 577F  FD7E19              ld a,(iy+lreml)
1162 5782  DD7707              ld (ix+ptol),a

1164 5785                      ;fall into WAIT
1165 5785
```

```
1166 5785                      ;from previous page

                       ;Wait 2**(LDBTIM-1) time and start again.

1170 5785  FD4616      wait:   ld b,(iy+ldbtim)
1171 5788  3E01                ld a,1                  ;compute 2**DBTIM-1
1172 578A  05                  dec b                   ;compute the DBTIM-1 par
1173 578B  2803                jr z,waitdb             ;2**0 is easy case
1174 578D  87          waitml: add a,a
1175 578E  10FD                djnz waitml

1177 5790  47          waitdb: ld b,a                  ;get counter to easy pla
1178 5791  CD4755      waitdl: call dbr                ;do the sleeping
1179 5794  10FB                djnz waitdl

                       ;Enter at WAITDC from DATA1B when we're timing out outst

1183 5796  CDDC57      waitdc: call timtgo             ;yes, check if time to g
1184 5799  CD0858              call cardrp             ;carrier going (if not,
1185 579C  CD3358              call brkcom             ;check for breaks, att
1186 579F  C3DA56              jp data1                ;no, loop back to DATA1

                       ;Enter data mode here at receiving end.  Perhaps a call

1191 57A2  FD7E0C      datar:  ld a,(iy+ltype1)        ;DCE with handshaking?
1192 57A5  E603                and shmsk+dcemsk
1193 57A7  FE03                cp shmsk+dcemsk
1194 57A9  2015                jr nz,dtndce

1196 57AB  061A                ld b,mdial              ;ring the equipment.
1197 57AD  CDE869              call mpcdo
1198 57B0  C403FC              call nz,rdie
1199 57B3  CDA369              call mpcuns

1201 57B6  FDCB15F6            set rngbit,(iy+lflag2)  ;set "ring has been init
1202 57BA  010A0C              ld bc,rngtim
1203 57BD  CDFE64              call settd              ;set timer

1205 57C0  FDCB0056    dtndce: bit hbit,(iy+ltype1)    ;a host?
1206 57C4  C2C556              jp nz,data0             ;yes, don't type messa

1208 57C7  21A45D              ld hl,remtxt            ;"connection from"
1209 57CA  CD9362              call strout
1210 57CD  FD4618              ld b,(iy+lremb)         ;load remote node
1211 57D0  FD4E19              ld c,(iy+lreml)         ; and line number.
1212 57D3  CD4762              call dstout
1213 57D6  CD9062              call crlf
1214 57D9  C3C556              jp data0                ;get to conversation.
1215 57DC
```

```
1216 57DC  FDCB1576   timtgo: bit rngbit,(iy+1flag2)  ;was ring initiated?
1217 57E0  2817               jr z,timt2              ;no, skip
1218 57E2  CDF764             call tdownp             ;has timer exhausted?
1219 57E5  2012               jr nz,timt2             ;no, skip
1220 57E7  FDCB15B6           res rngbit,(iy+1flag2)  ;kill ring bit
1221 57EB  CDCE6A             call mpcsta             ;check for carrier being
1222 57EE  CB77               bit carbit,a            ; established at timeout
1223 57F0  2007               jr nz,timt2             ;carrier did come up.
1224 57F2  FDCB15DE           set gbit,(iy+1flag2)    ;set "cleanly gone" bit.
1225 57F6  C3F05F             jp todead               ;kill the connection.

1227 57F9  FDCB155E   timt2:  bit gbit,(iy+1flag2)    ;carrier gone?
1228 57FD  C8                 ret z                   ;no, just return.
1229 57FE  CDEE6A             call mpckbc             ;keyboard run out?
1230 5801  CAF05F             jp z,todead             ;yes, break out right aw
1231 5804  CDF764             call tdownp             ;time out yet?
1232 5807  C0                 ret nz                  ;return if not...
1233 5808  C3F05F             jp todead               ;go to dead now that cou

1235 580B  CD3068     cardrp: call active             ;anything happen to lin
1236 580E  C8                 ret z                   ;return if not.

1238 580F  CDCE6A             call mpcsta             ;get MPC status.
1239 5812  FDCB1576           bit rngbit,(iy+1flag2)  ;was ring sent?
1240 5816  2808               jr z,card0              ;no, skip on
1241 5818  CB77               bit carbit,a            ;has carrier come on?
1242 581A  2804               jr z,card0              ;no, skip on
1243 581C  FDCB15B6           res rngbit,(iy+1flag2)  ;ring operation terminat

1245 5820  CB7F       card0:  bit gonbit,a            ;carrier drop?
1246 5822  C8                 ret z
1247 5823  FDCB15DE           set gbit,(iy+1flag2)    ;make sure everybody kn
1248 5827  CDEE6A             call mpckbc             ;get MPC KeyBoard Count.
1249 582A  CAF05F             jp z,todead             ;keyboard empty, go dire
1250 582D  010400             ld bc,dattim            ;load arbitrary time fo
1251 5830  C3FE64             jp settd                ;not empty, give it time
1252 5833
```

```
                        ;check for user BREAKing to CMD mode.  Also includes che
                        ;on dataset line.  Also handles sending breaks across ti

1256 5833  FDCB155E     brkcom: bit gbit,(iy+1flag2)    ;carrier gone?
1257 5837  C0                   ret nz                  ;yes, break may come, b
1258 5838  CD306B               call active             ;any activity on this li
1259 583B  C8                   ret z
1260 583C  CD7858               call gontst             ;test for dataset gone
1261 583F  FDCB0056             bit hbit,(iy+1type1)    ;is this a host link?
1262 5843  C0                   ret nz                  ;yes, can't BREAK to CMD
1263 5844  CDB96A               call mpcbka             ;read an clear break an
1264 5847  C8                   ret z

                        ;There is either a break or an attn, see what to do

1268 5848  113259               ld de,cmdbkp            ;in case we call yankme.
1269 584B  CB4F                 bit 1,a                 ;is it an attn char?
1270 584D  C28D64               jp nz,yankme            ;yes, to command mode
1271 5850  FDCB0076             bit pbkbit,(iy+1type1)  ;pass breaks?
1272 5854  CA8D64               jp z,yankme             ;no, to command mode

                        ;A break which we must pass

1276 5857  FD4618               ld b,(iy+1remb)         ;get destination address
1277 585A  FD4E19               ld c,(iy+1reml)
1278 585D  C5                   push bc
1279 585E  216C58               ld hl,xbreak            ;remote subr
1280 5861  E5                   push hl
1281 5862  CD0268               call rcall              ;send break
1282 5865  CD4255               call db1                ;wait 1 second for break
1283 5868  CDC15F               call clrbfi             ;then eat anything the u
1284 586B  C9                   ret                     ;ignore errors and retu

                        ;Remote subr to send in break.

1288 586C  0608         xbreak: ld b,msbk               ;send a break on the lin
1289 586E  CDE869               call mpcdo              ;do the function
1290 5871  C403FC               call nz,rdie            ; ++ Couldn't send brea
1291 5874  CDA369               call mpcuns
1292 5877  C9                   ret                     ;done

1294 5878  CDCE6A       gontst: call mpcsta             ;get MPC line status.
1295 587B  CB7F                 bit gonbit,a            ;is the line gone?
1296 587D  C8                   ret z                   ;no, return with status

1298 587E  FDCB15DE             set gbit,(iy+1flag2)    ;the host dropped carrie
1299 5882  C3F05F               jp todead               ;kill the line finally
1300 5885
```

```
                        ;SYNC will force the remote to report and reset its wind
                        ;SYNC sends off an RCALL to get local LOUT = remote LIN
                        ;is still full after that, it waits on second and then r
                        ;who presumably will call SYNC repeatedly until a window
                        ;SYNC used to block for the window to open but that mad
                        ;the caller to wath other events (like BRKCOM, etc.).
                        ;SYNC sets a flag called Window Wait (WWBIT) when
                        ;it is sleeping.  CSWIND pokes SYNC when the window op
                        ;This reduces SYNC's sleep to less than 1 second on the

1311 5885 FD4618        sync:   ld b,(iy+lremb)         ;push the destination
1312 5888 FD4E19                ld c,(iy+lreml)
1313 588B C5                    push bc
1314 588C 21A758                ld hl,xsync             ;push the remote subr
1315 588F E5                    push hl
1316 5890 CD0268                call rcall              ;call the remote version
1317 5893 E2E758                jp po,remerr            ;if can't reach it, gi

                        ;XSYNC returns A=remote's LIN.  It lies and returns 255

1321 5896 FD7718                ld (iy+lout),a          ;copy over remote LIN to
1322 5899 3C                    inc a                   ;is LOUT 255?
1323 589A C0                    ret nz                  ;no, return now
1324 589B FDCB14D6              set wwbit,(iy+lflag1)   ;set window wait bit for
1325 589F CD4255                call db1                ;here as long as in comm
1326 58A2 FDCB1496              res wwbit,(iy+lflag1)   ;clear window wait bit
1327 58A6 C9                    ret                     ;and return

                        ;This is the remote half of SYNC.  It returns A=LIN if

1331 58A7 FD7E1C        xsync:  ld a,(iy+lin)           ;return LIN, normall
1332 58AA FDCB1476              bit cmdbit,(iy+lflag1)  ;unless command mode
1333 58AE C8                    ret z                   ;if DATA, fine
1334 58AF 3EFF                  ld a,255                ;else, return 255
1335 58B1 FD361CFF              ld (iy+lin),255         ;and mark window as ful
1336 58B5 C9                    ret

                        ;Call HOLD when you want to force the remote's window t
                        ;If there is no remote, we return immediately.  The main
                        ;HOLD is the command processor which wants to keep the r
                        ;during commands.

1343 58B6 FDCB147E      hold:   bit cnvbit,(iy+lflag1)  ;is there one?
1344 58BA C8                    ret z                   ;no, all done!

1346 58BB FD4618                ld b,(iy+lremb)
1347 58BE FD4E19                ld c,(iy+lreml)
1348 58C1 C5                    push bc                 ;push destination
1349 58C2 21CE58                ld hl,xhold             ;and remote subr
1350 58C5 E5                    push hl
1351 58C6 CD0268                call rcall              ;poof
1352 58C9 FD361CFF              ld (iy+lin),255         ;force full window so RE
1353 58CD C9                    ret                     ;ignore errors and ret

                        ;Remote version.  Just plant a 255 in LOUT.

1357 58CE FD361BFF      xhold:  ld (iy+lout),255
```

```
1358 5802  C9                ret
1359 5803
```

```
                        ;NTAB indexed by line's RATE (0=50 baud, F=19.2kb) retur
                        ;characters we should have at that baud rate if we were
                        ;amount of time. Entries are computed in terms of DBR (1
                        ;DBMIN=1. If you change DBMIN, update this table!! Overf

1365 58D3  00000000     ntab:   defb    0,  0,  0,  0
1366 58D7  00010204             defb    0,  1,  2,  4
1367 58DB  06060810             defb    6,  6,  8, 12
1368 58DF  10182040             defb   16, 24, 32, 64
1369 58E3
```

```
                        ;Subroutine to check if remote is reachable.  If unreach
                        ;(goes to DEAD).  Uses AF.

1373 58E3  CD0559       remchk: call remup              ;is remote up - p?
1374 58E6  D8                   ret c                   ;if so, fine
1375 58E7  CD8864       remerr: call todedp

1377 58EA  0D0A         ncttxt: defb 13,10
1378 58EC  203F3130             defm ' ?100 Remote unreachable',0
1378 58F0  30205265
1378 58F4  6D6F7465
1378 58F8  20756E72
1378 58FC  65616368
1378 5900  61626C65
1378 5904  00

                        ;Subroutine to determine whether node number (in A) is
                        ;Carry if reachable, NC if not.  Uses AF only.  Enter at
                        ;asking about your remote.

1384 5905  FD7E18       remup:  ld a,(iy+lremb)         ;load box number from li

1386 5908  E5           nodeup: push hl                 ;save some temps
1387 5909  C5                  push bc
1388 590A  21FB42               ld hl,costs             ;get cost table entry
1389 590D  0600                 ld b,0
1390 590F  4F                   ld c,a
1391 5910  09                   add hl,bc
1392 5911  7E                   ld a,(hl)
1393 5912  C1                   pop bc
1394 5913  E1                   pop hl
1395 5914  C601                 add a,1                 ;if it was FF, make it
1396 5916  3F                   ccf                     ;clear C if it was FF, e
1397 5917  C9                   ret
1398 5918
```

```
                        ;This is the CMD state. Initially, we HOLD the other end
                        ;one) so the user doesn't get garbage interspersed with
                        ;input-output. We then process commands. It has two entr
                        ;from DATA state on BREAK which should print herald.

1404 5918  FDCB14F6     cmd:    set cmdbit,(iy+1flag1)  ;put us in command mode
1405 591C  CD785F               call cmdchk             ;consistency check comma
1406 591F  CDB658               call hold               ;hold the remote
1407 5922  CD985F               call chrin              ;get the character that

1409 5925  FE03                 cp ctrlc                ;control-C?
1410 5927  CADF5A               jp z,upc                ;yes, skip messages, etc

1412 592A  114056               ld de,dead
1413 592D  FE0D                 cp ctrlm                ;permit only one entry t
1414 592F  C48D64               call nz,yankme          ; a carriage return.

1416 5932  FDCB14F6     cmdbkp: set cmdbit,(iy+1flag1)  ;set command mode if ent
1417 5936  CD785F               call cmdchk             ;consistency-check comma
1418 5939  CDB658               call hold               ;hold the remote
1419 593C  0605                 ld b,5                  ;wait .5 seconds
1420 593E  CD4C55       cmdbk1: call db01
1421 5941  10FB                 djnz cmdbk1
1422 5943  CDC15F               call clrbfi             ;then clear input buffer
1423 5946  210342               ld hl,name0             ;(enter here from DATA's
1424 5949  CD9362               call strout
1425 594C  210D59               ld hl,vertxt            ;output version number
1426 594F  CD9362               call strout
1427 5952  ED4B0142             ld bc,(vers)            ;get our version
1428 5956  CD4762               call dstout             ;type it as destination
1429 5959  CD9062               call crlf
1430 595C  1810                 jr next                 ;start processing comma
1431 595E
```

```
                        ;There are a few main labels for the command processor.

                        ;ERRORP         pop error message address from stack int
                        ;ERROR          type error message in HL and fix stack,
                        ;ERRORQ         just fix stack, etc. (q=quiet)
                        ;ERRORX         type "standard" error message fix stack,
                        ;NEXT           type crlf and prompt for next command

                        ;In addition, there are some very useful subroutines.
                        ;Some of them go directly to ERROR or ERRORQ if they det

                        ;CHRIN          get a character without echoing it
                        ;CMDIN          get character echoing as appropriate, re
                        ;CHROUT         print a character fixing parity, etc.
                        ;STROUT         print a string of characters.
                        ;CRLF           print a crlf
                        ;DECIN          read decimal number into C, delim into
                        ;DECINC         just like DECIN except MUST have number
                        ;DECOUT         write decimal number from C
                        ;DEC3R          write 3-digit, right-justified decimal
                        ;DEC3L          write 3-digit, left-justified decimal nu
                        ;CNFIRM         prompt [confirm] and wait for crlf, erro
                        ;CRCHEK         check character in A against CR, error
                        ;DSTIN          get a destination into BC, error if not
                        ;DSTOUT         print a destination from BC
                        ;DSTFOU         print 7 character destination (nnn.mmm)
                        ;ECHIN          echo input character, convert lower case
                        ;ECHINL         echo input character.
                        ;IMGIN          get a raw character, no echo, no parity

1462 595E  E1           errorp: pop hl                  ;string address on the s

1464 595F  CD9362       error:  call strout             ;type the error string
1465 5962  CD9D62               call crlf

1467 5965  CDC15F       errorq: call clrbfi             ;clear input typeahead
1468 5968  117559               ld de,nextq             ;reset pdl, continue.
1469 596B  CD8D64               call yankme
1470 596E
```

```
                        ;Here to process a new command. NEXT prints CRLF, NEXTQ
1473 596E CD9062        next:   call crlf
1474 5971 FDCB15AE              res hedbit,(iy+1flag2)  ;clear one-time bit for

1476 5975 CD1264        nextq:  call givlok             ;give up the command int
1477 5978 3A0042                ld a,(me)              ;get our node number
1478 597B 47                    ld b,a                 ;into B
1479 597C FD4E0D                ld c,(iy+1num)         ;and link number
1480 597F CD4762                call dstout            ;type this destination
1481 5982 FDCB147E              bit cnvbit,(iy+1flag1) ;in a connection?
1482 5986 280F                  jr z,next2             ;if not, skip second par
1483 5988 21E459                ld hl,totxt            ;get the " to " phrase
1484 598B CD9362                call strout
1485 598E FD4618                ld b,(iy+1remb)        ;remote box
1486 5991 FD4E19                ld c,(iy+1rem1)        ;remote link
1487 5994 CD4762                call dstout            ;type it

1489 5997 21E959        next2:  ld hl,pmttxt           ;get the prompt string
1490 599A CD9362                call strout            ;output it

1492 599D 21EE59                ld hl,cm0tab           ;get pointer to command
1493 59A0 1801                  jr disptc              ;dispatch

                        ;Enter here to move down one command level.  Input chara
                        ;command table in HL.

1498 59A2 E1            dispth: pop hl                 ;get pointer to command
1499 59A3 CDF65F        disptc: call echin             ;get a character, with
1500 59A6 CD1363                call abort             ;an abort of some sort?
1501 59A9 FE20                  cp ' '                 ;space?
1502 59AB 28F6                  jr z,disptc            ;yes, ignore it
1503 59AD FE3F                  cp '?'                 ;question mark?
1504 59AF CAB55D                jp z,quest             ;yup, display command ta
1505 59B2 54                    ld d,h                 ;save base pointer in D
1506 59B3 5D                    ld e,l

1508 59B4 BE            dislop: cp (hl)                ;look at first byte
1509 59B5 23                    inc hl                 ;step to first byte of d
1510 59B6 2810                  jr z,match             ;got a match
1511 59B8 23                    inc hl                 ;skip remaining part of
1512 59B9 23                    inc hl
1513 59BA 23                    inc hl
1514 59BB 23                    inc hl
1515 59BC CB7E                  bit 7,(hl)             ;negative?
1516 59BE 28F4                  jr z,dislop            ;no, keep looping
1517 59C0 3E07                  ld a,ctrlg             ;end of table, type BEL
1518 59C2 CDA05F                call chrout
1519 59C5 EB                    ex de,hl               ;get HL back
1520 59C6 18DB                  jr disptc

1522 59C8 4E            match:  ld c,(hl)              ;push next word onto sta
1523 59C9 23                    inc hl
1524 59CA 46                    ld b,(hl)
1525 59CB 23                    inc hl
1526 59CC C5                    push bc                ;so we can jump there s
1527 59CD 7E                    ld a,(hl)              ;get address of string
```

```
1528 59CE  23          inc hl
1529 59CF  66          ld h,(hl)
1530 59D0  6F          ld l,a          ;into HL
1531 59D1  CD9362      call strout     ;type it
1532 59D4  C9          ret             ;jump into routine speci
1533 59D5
```

```
                        ;Text and tables for previous page

1536 59D5  2C2D2C56     vertxt: defm ' - Version ',0
1536 59D9  65727369
1536 59DD  6F6E2C00
1537 59E1  0D0A         crltxt: defb ctrlm,ctrlj
1538 59E3  00           nultxt: defb 0

1540 59E4  2C746F20     totxt:  defm ' to ',0
1540 59E8  00
1541 59E9  2C2D2D20     pmttxt: defm ' -- ',0
1541 59ED  00

                        ;Dispatch table for level 0 commands

1545 59EE  41           cm0tab: defb 'A'
1546 59EF  275B2C5B             defw acc,acctxt
1547 59F3  43                   defb 'C'
1548 59F4  325B2A5B             defw cnc,cnctxt
1549 59F8  44                   defb 'D'
1550 59F9  E65DDC5D             defw dsc,dsctxt
1551 59FD  48                   defb 'K'
1552 59FE  055E015E             defw kil,kiltxt
1553 5A02  4D                   defb 'M'
1554 5A03  245A215A             defw map,maptxt
1555 5A07  52                   defb 'R'
1556 5A08  F85DF25D             defw rsm,rsmtxt
1557 5A0C  53                   defb 'S'
1558 5A0D  105E0C5E             defw sit,sittxt
1559 5A11  21                   defb '!'
1560 5A12  725FE359             defw xcl,nultxt
1561 5A16  0D                   defb ctrlm
1562 5A17  6E59E359             defw next,nultxt
1563 5A1B  03                   defb ctrlc
1564 5A1C  DF5AE359             defw upc,nultxt
1565 5A20  FF                   defb -1
1566 5A21
```

```
                        ;Map network

1569 5A21  617000       maptxt: defm 'ap',0

1571 5A24  CDC562       map:    call cnfirm             ;get confirmation
1572 5A27  CD9062               call crlf
1573 5A2A  CD9062               call crlf

1575 5A2D  0600                 ld b,0                  ;start at node 0
1576 5A2F  CD3E5A       maplop: call mapnod             ;map this node
1577 5A32  04                   inc b                   ;step to next one
1578 5A33  78                   ld a,b
1579 5A34  FE10                 cp nbox
1580 5A36  38F7                 jr c,maplop             ;until done
1581 5A38  CD9062               call crlf
1582 5A3B  C36E59               jp next
1583 5A3E
```

```
                        ;Subroutine to MAP one node's worth.  Enter with Node nu
1586 5A3E 0E00          mapnod: ld c,0                  ;start with link 0
1587 5A40 C5            mapnlp: push bc                 ;save destination
1588 5A41 219F5A                ld hl,xmap
1589 5A44 E5                    push hl
1590 5A45 CD0268                call rcall              ;do MAP for links starti
1591 5A48 E0                    ret po                  ;if RCALL fails, skip t
1592 5A49 D0                    ret nc                  ;if XMAP says none, skip
1593 5A4A CD505A                call maplnk             ;map this link
1594 5A4D 0C                    inc c                   ;start at one following
1595 5A4E 18F0                  jr mapnlp               ;then use XMAP's BC to r

                        ;Subroutine to MAP one link's worth.  Call this after c
                        ;that is, BC=this link, DE=destination, A=flags, H=host,

1600 5A50 C5            maplnk: push bc                 ;save link number
1601 5A51 CD6D5A                call mapout             ;output this info
1602 5A54 CD8A62                call stroth
1603 5A57 202D2D00              defm ' - ',0
1604 5A5B 42                    ld b,d
1605 5A5C 4B                    ld c,e                  ;then do other side
1606 5A5D C5                    push bc
1607 5A5E 21C45A                ld hl,xmap2             ;just get info
1608 5A61 E5                    push hl
1609 5A62 CD0268                call rcall
1610 5A65 EC6D5A                call pe,mapout
1611 5A68 CD9062                call crlf
1612 5A6B C1                    pop bc
1613 5A6C C9                    ret

1615 5A6D F5            mapout: push af                 ;save flags
1616 5A6E CD5E62                call dstfou             ;output BC
1617 5A71 CD9E5F                call blank
1618 5A74 F1                    pop af                  ;what kind?
1619 5A75 CB67                  bit vbit,a
1620 5A77 2019                  jr nz,maplv             ;VLINK
1621 5A79 CB57                  bit hbit,a
1622 5A7B 3E48                  ld a,'H'                ;assume HLINK
1623 5A7D 2002                  jr nz,mapl2             ;guessed right
1624 5A7F 3E54                  ld a,'T'
1625 5A81 CDA05F        mapl2:  call chrout             ;output it
1626 5A84 4C                    ld c,h                  ;get host number
1627 5A85 CD906C                call dec3l              ;output it
1628 5A88 3E40          mapl3:  ld a,'@'
1629 5A8A CDA05F                call chrout
1630 5A8D 7D                    ld a,l                  ;get rate
1631 5A8E CD0661                call ratout
1632 5A91 C9                    ret

1634 5A92 E5            maplv:  push hl                 ;save HL
1635 5A93 CD8A62                call stroth
1636 5A96 56202020              defm 'V   ',0
1636 5A9A 00
1637 5A9B E1                    pop hl
1638 5A9C 18EA                  jr mapl3
1639 5A9E
```

```
                        ;Remote subroutine XMAP.  Called with BC=first link to l
                        ;with BC=first one which matches, or returns NC if none
                        ;if C is .GE. NLINK.  If a match is returned, DE=other e
                        ;H=local host and L=local speed.  This subroutine will o
                        ;matches where BC is .LT. DE, that is, this is the lowe
                        ;of the connection.  This causes mathes to be printed on
                        ;Enter at XMAP2 if you already have BC pointing to the w

1648 5A9E  0C           xmapn:  inc c
1649 5A9F  79           xmap:   ld a,c                  ;at the end?
1650 5AA0  FE20                 cp nlink
1651 5AA2  3039                 jr nc,xmapf             ;yes - fail
1652 5AA4  CDDE64               call iylin              ;point there
1653 5AA7  FDCB005E             bit onbit,(iy+ltype1)   ;on-line?
1654 5AAB  20F1                 jr nz,xmapn             ;no, skip it
1655 5AAD  3E80                 ld a,cnvmsk             ;assume we're checking f
1656 5AAF  FDCB0066             bit vbit,(iy+ltype1)    ;VLINK?
1657 5AB3  2802                 jr z,xmapt              ;no, we guess the correc
1658 5AB5  3E08                 ld a,vupmsk             ;guessed wrong - try thi
1659 5AB7  FDA614       xmapt:  and (iy+lflag1)         ;check flags
1660 5ABA  28E2                 jr z,xmapn              ;didn't find bit - PUNT!

1662 5ABC  FD7E18       xmapy:  ld a,(iy+lremb)         ;is B .LT. REMB?
1663 5ABF  B8                   cp b
1664 5AC0  2813                 jr z,xmape              ;if B .EQ. REMB, check R
1665 5AC2  38DA                 jr c,xmapn              ;if B. GT. REMB, skip t

1667 5AC4  FD5618       xmap2:  ld d,(iy+lremb)
1668 5AC7  FD5E19               ld e,(iy+lreml)
1669 5ACA  FD6604               ld h,(iy+lhost)
1670 5ACD  FD6E0A               ld l,(iy+lrate)
1671 5AD0  FD7E14               ld a,(iy+lflag1)
1672 5AD3  37                   scf
1673 5AD4  C9                   ret

1675 5AD5  FD7E19       xmape:  ld a,(iy+lreml)
1676 5AD8  B9                   cp c
1677 5AD9  38C3                 jr c,xmapn              ;if REML .LT. C, skip t
1678 5ADB  18E7                 jr xmap2                ;else do it

1680 5ADD  AF           xmapf:  xor a                   ;clear carry
1681 5ADE  C9                   ret
1682 5ADF
```

```
                        ;Connect via control-C (to default host).

1685 5ADF  FDCB016E     upc:    bit autbit,(iy+ltype2)  ;auto-connect enabled?
1686 5AE3  2025                 jr nz,upc0              ;yup, proceed.
1687 5AE5  CD5E59               call errorp
1688 5AE8  203F3030             defm ' ?001 Auto-connect not configured',0
1688 5AEC  31204175
1688 5AF0  746F2D63
1688 5AF4  6F6E6E65
1688 5AF8  63742D6E
1688 5AFC  6F742063
1688 5B00  6F6E6669
1688 5B04  67757265
1688 5B08  6400

1690 5B0A  CD6863       upc0:   call dscchk
1691 5B0D  CD1E64               call discon             ;address appears valid,
1692 5B10  37                   scf                     ;want to get into data m
1693 5B11  F5                   push af                 ; should connection succ
1694 5B12  FD4605               ld b,(iy+lautoh)        ;load high byte of auto
1695 5B15  FD4E06               ld c,(iy+lautol)        ; and low byte
1696 5B18  78                   ld a,b                  ;load code.
1697 5B19  FEFF                 cp 0ffh                 ;to a host?
1698 5B1B  CAD35B               jp z,cncho              ;get into "connect to ho
1699 5B1E  1840                 jr cncadr               ; conversation and try f
1700 5B20
```

```
                        ;Access

1703 5B20  63636573     acctxt: defm 'ccess ',0
1703 5B24  732000
1705 5B27  AF           acc:    xor a                   ;indicate desire to stay
1706 5B28  1809                 jr cncdo                ;get into common connect

                        ;Connect

1710 5B2A  6F6E6E65     cnctxt: defm 'onnect ',0
1710 5B2E  63742000
1712 5B32  37           cnc:    scf                     ;indicate desire to en
1713 5B33  F5           cncdo:  push af                 ;store flags
1714 5B34  CDEC61               call dstin              ;get a destination in BC
1715 5B37                       ;DSTIN checks delim!
1716 5B37  FDCB147E             bit cnvbit,(iy+1flag1)  ;in a connection?
1717 5B3B  281B                 jr z,cncdsc             ;no, skip this
1719 5B3D  C5                   push bc
1720 5B3E  CD8A62               call stroth             ;Tell user we're disconn
1721 5B41  2D2D2D20             defm ' -- Disconnecting ',0
1721 5B45  44697363
1721 5B49  6F6E6E65
1721 5B4D  6374696E
1721 5B51  672000
1722 5B54  CD1E64               call discon             ;remove any previous con
1723 5B57  C1                   pop bc
1725 5B58  CD4255       cncdsc: call db1                ;wait 1 second so billin
1726 5B5B  78                   ld a,b                  ;which type of connect?
1727 5B5C  3C                   inc a
1728 5B5D  CAD35B               jp z,cncho              ;if B=FF then host, else
1730 5B60  C5           cncadr: push bc                 ;save destination
1731 5B61  C5                   push bc                 ;put destination on stac
1732 5B62  215F5C               ld hl,xcnadr            ;connect to specific ad
1733 5B65  E5                   push hl
1734 5B66  CD7969               call gcid               ;get a sequence into A
1735 5B69  FD7717               ld (iy+1cid),a          ;store it locally
1736 5B6C  5F                   ld e,a                  ;pass it to XCNADR throu
1737 5B6D  FD5607               ld d,(iy+logrp)         ;and originate group in
1738 5B70  FDCB14FE             set cnvbit,(iy+1flag1)  ;pretend we're already
1739 5B74                       ;connection so that the
1740 5B74                       ;from the remote looks g
1741 5B74  CD0268               call rcall              ;try to connect there
1742 5B77  C1                   pop bc                  ;get our saved copy of d
1743 5B78                       ;note: CNVBIT is on at t
1744 5B78                       ;must clear it if any e
1745 5B78  E2C95B               jp po,netrbl            ;if RCALL fails, give ne
1746 5B7B  304F                 jr nc,cncluz            ;else remote subr has fa
1747 5B7D
```

```
                        ;connection won, put BC into LREMB/L

1750 5B7D  FD7018       cncwon: ld (iy+lremb),b
1751 5B80  FD7119               ld (iy+lreml),c
1752 5B83  FD341A               inc (iy+lwatch)         ;so WATCH will see it
1753 5B86  CD8A62               call stroth
1754 5B89  2D2D2D20             defm ' -- Connected via ',0
1754 5B8D  436F6E6E
1754 5B91  65637465
1754 5B95  64207669
1754 5B99  612000
1755 5B9C  CD4762               call dstout
1756 5B9F  F1                   pop af                  ;see if user wants to st
1757 5BA0  D26E59               jp nc,next              ;yes, go get next comm
1758 5BA3  CD9062       todata: call crlf               ;clean up line.
1759 5BA6  FDCB14B6             res cmdbit,(iy+lflag1)  ;done with command mode
1760 5BAA  11C556               ld de,data0             ;put us in DATA state
1761 5BAD  CD8D64               call yankme

1763 5BB0  CD5E59       netgon: call errorp             ;net trouble error mes
1764 5BB3  203F3030     tbltxt: defm ' ?002 Network trouble',0
1764 5BB7  32204E65
1764 5BBB  74776F72
1764 5BBF  68207472
1764 5BC3  6F75626C
1764 5BC7  6500

1766 5BC9  21B35B       netrbl: ld hl,tbltxt            ;enter here if remote no
1767 5BCC  FDCB14BE     cncluz: res cnvbit,(iy+lflag1)  ;enter here with remot
1768 5BD0  C35F59               jp error                ;HL also has some error
1769 5BD3
```

```
                        ;Here when BC has FF.host in it. Try ME first, then each

1772 5BD3  3A0042       cncho:  ld a,(me)               ;try ME first
1773 5BD6  CD265C               call cnctry
1774 5BD9  38A2                 jr c,cncwon             ;we won..

1776 5BDB  FD7118               ld (iy+lremb),c         ;save host number.

                        ;the connect logic has the following register assignment

                        ;       B       current node under test
                        ;       C       floor of cost
                        ;       D       best node found so far (lowest cost)
                        ;       E       cost of best node found so far

                        ;in general, we will search for a node of minimum cost w
                        ;than C.  When a node of this cost is found, it, and th
                        ;same cost, will be checked for possible connection.  If
                        ;cost used will go to E and the process repeated.

1790 5BDE  0E00                 ld c,0                  ;load floor of cost (cos

1792 5BE0  0600         cncl0:  ld b,0                  ;set current node to te
1793 5BE2  11FFFF               ld de,0ffffh            ;load "best" cost found
1794 5BE5                       ;                       ; and the node to blame
1795 5BE5  21F842               ld hl,costs             ;point to COSTS(B).

1797 5BE8  7E           cncl1:  ld a,(hl)               ;load COSTS(B).
1798 5BE9  B8                   cp e                    ;better than last "best"
1799 5BEA  3008                 jr nc,cncl2             ;this node is no good.

1801 5BEC  B9                   cp c                    ;is the cost greater th
1802 5BED  FAF45B               jp m,cncl2              ;skip it if it is less
1803 5BF0  2802                 jr z,cncl2              ; than or equal to floor

1805 5BF2  5F                   ld e,a                  ;copy this as "best cost
1806 5BF3  50                   ld d,b                  ; and remember box numbe

1808 5BF4  04           cncl2:  inc b                   ;get to next box and
1809 5BF5  23                   inc hl                  ; point to its cost.
1810 5BF6  78                   ld a,b
1811 5BF7  FE10                 cp nbox
1812 5BF9  38ED                 jr c,cncl1

1814 5BFB  21D25C               ld hl,bsytxt
1815 5BFE  7A                   ld a,d                  ;get "best" node number.
1816 5BFF  FEFF                 cp 0ffh                 ;was one found?
1817 5C01  CACC5B               jp z,cncluz             ;nope, quit.
1818 5C04
```

```
                        ;a node was found, attempt connection.

1821 5C04  D5           cncl3:  push de                 ;save node, cost in D, E
1822 5C05  7A                   ld a,d                  ;send to this node.
1823 5C06  FD4E18               ld c,(iy+1remb)         ;get host number into B
1824 5C09  CD265C               call cnctry             ;try the connection
1825 5C0C  D1                   pop de                  ;restore node (D) and it
1826 5C0D  DA7D5B               jp c,cncwon             ;jump if connection re

1828 5C10  14                   inc d                   ;try next node for equal
1829 5C11  4A                   ld c,d                  ;copy node number.
1830 5C12  0600                 ld b,0
1831 5C14  21FB42               ld hl,costs
1832 5C17  09                   add hl,bc               ;index to COSTS(D).
1833 5C18  4B                   ld c,e                  ;must have cost in C if

1835 5C19  7A           cncl4:  ld a,d
1836 5C1A  FE10                 cp nbox                 ;if no more nodes to che
1837 5C1C  30C2                 jr nc,cncl0             ; cost, then start loop

1839 5C1E  7E                   ld a,(hl)               ;load cost of this node
1840 5C1F  B9                   cp c                    ;if it has the same cost
1841 5C20  28E2                 jr z,cncl3              ; last node, try it as

1843 5C22  14                   inc d                   ;step to next box
1844 5C23  23                   inc hl                  ; (and its cost).
1845 5C24  18F3                 jr cncl4
1846 5C26
```

```
                        ;subroutine CNCTRY takes destination in B(FF).C and node
                        ;an RCALL of XCNHO at the remote.  Returns C with winni
                        ;preserves it.  Uses HL.

1851 5C26  67           cnctry: ld h,a                  ;send this to A.0
1852 5C27  2E00                 ld l,0
1853 5C29  E5                   push hl
1854 5C2A  21445C               ld hl,xcnho             ;tell it to connect to
1855 5C2D  E5                   push hl
1856 5C2E  FD5E17               ld e,(iy+lcid)          ;get the sequence
1857 5C31  FD5607               ld d,(iy+logrp)         ;and orig group
1858 5C34  FDCB14FE             set cnvbit,(iy+lflag1)  ;see comment at CNCADR
1859 5C38  CD0268               call rcall
1860 5C3B  E23F5C               jp po,tryluz            ;if unreachable
1861 5C3E  D8                   ret c                   ;else return caller and

1863 5C3F  FDCB14BE     tryluz: res cnvbit,(iy+lflag1)
1864 5C43  C9                   ret                     ;see coments at CNCADR

                        ;Remote subr to connect to a host in C (B=FF). Enter wi
                        ;host in C, Orig GRP in D. Returns C/NC. If C, BC=addres

1869 5C44  FD211B43     xcnho:  ld iy,lblk0             ;start at line 0

1871 5C48  FDCB0056     xcnlop: bit hbit,(iy+ltype1)    ;an H-link?
1872 5C4C  280B                 jr z,xcnnxt
1873 5C4E  FD7E04               ld a,(iy+lhost)         ;reasonable host?
1874 5C51  B9                   cp c
1875 5C52  2005                 jr nz,xcnnxt

1877 5C54  CD5F5C               call xcnadr             ;try to connect here
1878 5C57  384D                 jr c,xcnwon             ;if XCNADR says okay,

1880 5C59  CDD664       xcnnxt: call iynext             ;else try next
1881 5C5C  38EA                 jr c,xcnlop
1882 5C5E  C9                   ret                     ;return with carry off
1883 5C5F
```

```
                        ;Subroutine to connect to line in IY, sequence in E, OGR
                        ;Returns C/NC, if NC, HL has error code.

1887 5C5F FD7E14        xcnadr: ld a,(iy+lflag1)        ;load line flags (LFLAG)

1889 5C62 21D25C                ld hl,cnetxt            ;connected?
1890 5C65 CB7F                  bit cnvbit,a            ;check LFLAG
1891 5C67 2067                  jr nz,xcnerr

1893 5C69 21225D                ld hl,cmetxt            ;command-ing?
1894 5C6C CB77                  bit cmdbit,a            ;check LFLAG
1895 5C6E 2060                  jr nz,xcnerr

1897 5C70 FD7E00                ld a,(iy+ltype1)        ;get line type bits (LT

1899 5C73 21865D                ld hl,ofetxt            ;off
1900 5C76 CB5F                  bit onbit,a             ;check LTYPE
1901 5C78 2856                  jr z,xcnerr

1903 5C7A 21685D                ld hl,vletxt            ;V-link?
1904 5C7D CB67                  bit vbit,a              ;check LTYPE
1905 5C7F 204F                  jr nz,xcnerr

1907 5C81 21E65C                ld hl,etdtxt            ;dataset?
1908 5C84 CB4F                  bit shbit,a             ;check LTYPE
1909 5C86 2804                  jr z,xcnnds             ;make sure not a full
1910 5C88 CB47                  bit dcebit,a
1911 5C8A 2844                  jr z,xcnerr

1913 5C8C CDDE6A        xcnnds: call mpcrat             ;get the rate
1914 5C8F 21055D                ld hl,hnetxt            ;hunt error?
1915 5C92 FEFF                  cp bhunt
1916 5C94 283A                  jr z,xcnerr

1918 5C96 FD7E08                ld a,(iy+lagrp)         ;load access group
1919 5C99 B7                    or a                    ;is it zero?
1920 5C9A 280A                  jr z,xcnwon             ;yes, let anybody connec

1922 5C9C A2                    and d                   ;else, do groups interse
1923 5C9D 2007                  jr nz,xcnwon            ;bits match, ok...
1924 5C9F 214A5D                ld hl,gretxt            ;prepare error message
1925 5CA2 CB7A                  bit g7bit,d             ;is priv bit on?
1926 5CA4 282A                  jr z,xcnerr             ;no, loser...
1927 5CA6
```

```
                        ;Here when we like the connection. Reach into the SUBR p
                        ;IX) to find the sender and put us in the connection. A
                        ;to destination it used.

1932 5CA6  FD7317       xcnwon: ld (iy+lcid),e          ;set sequence
1933 5CA9  DD7E08               ld a,(ix+pfrb)          ;get from
1934 5CAC  E67F                 and low7
1935 5CAE  47                   ld b,a                  ;copy node number.
1936 5CAF  FD7718               ld (iy+lremb),a
1937 5CB2  DD7E09               ld a,(ix+pfrl)
1938 5CB5  E67F                 and low7
1939 5CB7  4F                   ld c,a                  ;copy link number.
1940 5CB8  FD7719               ld (iy+lreml),a
1941 5CBB  FDCB14FE             set cnvbit,(iy+lflag1)
1942 5CBF  D5                   push de
1943 5CC0  11A257               ld de,datar             ;put him in data
1944 5CC3  CDA164               call yankhm
1945 5CC6  D1                   pop de
1946 5CC7  3A0042               ld a,(me)               ;get our address into HL
1947 5CCA  47                   ld b,a                  ;set B=ME
1948 5CCB  FD4E0C               ld c,(iy+lnum)
1949 5CCE  37                   scf                     ;carry
1950 5CCF  C9                   ret

                        ;Here when we want to bomb caller. Error message in HL.

1954 5CD0               ccret:                          ;Clear-Carry RETurn.
1955 5CD0  AF           xcnerr: xor a                   ;clear carry
1956 5CD1  C9                   ret                     ;return
1957 5CD2
```

```
1958 5CD2                bsytxt:
1959 5CD2  203F3030      cnetxt: defm ' ?006 Not available',0
1959 5CD6  36204E6F
1959 5CDA  74206176
1959 5CDE  61696C61
1959 5CE2  626C6500
1960 5CE6  203F3030      etdtxt: defm ' ?003 Destination is a dataset',0
1960 5CEA  33204465
1960 5CEE  7374696E
1960 5CF2  6174696F
1960 5CF6  6E206973
1960 5CFA  20612064
1960 5CFE  61746173
1960 5D02  657400
1961 5D05  203F3030      hnetxt: defm ' ?005 Destination is hunting',0
1961 5D09  35204465
1961 5D0D  7374696E
1961 5D11  6174696F
1961 5D15  6E206973
1961 5D19  2068756E
1961 5D1D  74696E67
1961 5D21  00
1962 5D22  203F3030      cmetxt: defm ' ?007 Destination is entering a command',0
1962 5D26  37204465
1962 5D2A  7374696E
1962 5D2E  6174696F
1962 5D32  6E206973
1962 5D36  20656E74
1962 5D3A  6572696E
1962 5D3E  67206120
1962 5D42  636F6D6D
1962 5D46  616E6400
1963 5D4A  203F3030      gretxt: defm ' ?008 Administratively denied',0
1963 5D4E  38204164
1963 5D52  6D696E69
1963 5D56  73747261
1963 5D5A  74697665
1963 5D5E  6C792064
1963 5D62  656E6965
1963 5D66  6400
1964 5D68  203F3030      vletxt: defm ' ?004 Destination is a V-Link',0
1964 5D6C  34204465
1964 5D70  7374696E
1964 5D74  6174696F
1964 5D78  6E206973
1964 5D7C  20612056
1964 5D80  2D4C696E
1964 5D84  6B00
1965 5D86  203F3030      ofetxt: defm ' ?009 Destination is Off-Line',0
1965 5D8A  39204465
1965 5D8E  7374696E
1965 5D92  6174696F
1965 5D96  6E206973
1965 5D9A  204F6666
1965 5D9E  2D4C696E
1965 5DA2  6500
1966 5DA4  436F6E6E      remtxt: defm 'Connection from ',0
```

```
1966 5DA8  65637469
1966 5DAC  6F6E2066
1966 5DB0  726F6D20
1966 5DB4  00
1967 5DB5
```

```
                        ;question mark command (display the command list).

1970 5DB5  E5           quest:  push hl                 ;store pointer to comman
1971 5DB6  CD9062       questl: call crlf               ;write a return/line fee
1972 5DB9  E1                   pop hl                  ;restore pointer.
1973 5DBA  7E           questq: ld a,(hl)               ;load first character.
1974 5DBB  B7                   or a                    ;test it.
1975 5DBC  FA6E59               jp m,next               ;return if list exhaust
1976 5DBF  FE20                 cp ' '                  ;printable?
1977 5DC1  3812                 jr c,quests
1978 5DC3  CDA05F               call chrout             ;write the character.
1979 5DC6  23                   inc hl                  ;skip character.
1980 5DC7  23                   inc hl                  ;skip subroutine call ad
1981 5DC8  23                   inc hl                  ; ...
1982 5DC9  4E                   ld c,(hl)               ;load string address
1983 5DCA  23                   inc hl
1984 5DCB  46                   ld b,(hl)               ; into BC.
1985 5DCC  23                   inc hl                  ;point to next command c
1986 5DCD  E5                   push hl                 ;save pointer.
1987 5DCE  C5                   push bc                 ;push string address
1988 5DCF  E1                   pop hl                  ; and move it to HL
1989 5DD0  CD9362               call strout             ; for the string typing
1990 5DD3  18E1                 jr questl

1992 5DD5  23           quests: inc hl                  ;skip to next command ch
1993 5DD6  23                   inc hl
1994 5DD7  23                   inc hl
1995 5DD8  23                   inc hl
1996 5DD9  23                   inc hl
1997 5DDA  18DE                 jr questq               ;skip it quietly.
1998 5DDC
```

```
                        ;Disconnect

2001 5DDC  6973636F     dsctxt: defm 'isconnect',0
2001 5DE0  6E6E6563
2001 5DE4  7400

2003 5DE6  CD4F63       dsc:    call cnchek             ;must be in command
2004 5DE9  CDC562               call cnfirm
2005 5DEC  CD1E64               call discon
2006 5DEF  C36E59               jp next

                        ;Resume

2010 5DF2  65737560     rsmtxt: defm 'esume',0
2010 5DF6  6500

2012 5DF8  CD4F63       rsm:    call cnchek
2013 5DFB  CDC562               call cnfirm
2014 5DFE  C3A358               jp todata

                        ;Kill

2018 5E01  696C6C00     kiltxt: defm 'ill',0

2020 5E05  CDC562       kil:    call cnfirm
2021 5E08  CD8864               call todedp             ;go to DEAD,
2022 5E0B  00                   defb 0                  ; type nul text.
2023 5E0C
```

```
                        ;Set

                        ;These commands are rather tricky to follow. They almost
                        ;RCALLs to perform some action to some link somewhere. T
                        ;on the top of the stack (initially set to this link).
                        ;"remote" modifier.

2031 5E0C  65742000     sittxt: defm 'et ',0

2033 5E10  3A0042       sit:    ld a,(me)               ;get our own destination
2034 5E13  47                   ld b,a
2035 5E14  FD4E0D               ld c,(iy+lnum)
2036 5E17  C5                   push bc                 ;prime stack with it
2037 5E18  CDA259       sitlop: call dispth
2038 5E18  45                   defb 'E'
2039 5E1C  3E5F4E5E             defw siteco,ecotxt
2040 5E20  46                   defb 'F'
2041 5E21  6C5E375E             defw sitflo,sfltxt
2042 5E25  49                   defb 'I'
2043 5E26  C85E445E             defw sitint,sintxt
2044 5E2A  52                   defb 'R'
2045 5E2B  535E3C5E             defw sitrem,srmtxt
2046 5E2F  FF                   defb -1

2048 5E30  65606F74     srmtxt: defm 'emote ',0
2048 5E34  652000
2049 5E37  6C6F7720     sfltxt: defm 'low control ',0
2049 5E38  636F6E74
2049 5E3F  726F6C20
2049 5E43  00
2050 5E44  6E746572     sintxt: defm 'nterrupt ',0
2050 5E48  72757074
2050 5E4C  2000
2051 5E4E  63686F20     ecotxt: defm 'cho ',0
2051 5E52  00


                        ;Set Remote

2056 5E53  CD4F63       sitrem: call cnchek             ;in a connection?
2057 5E56  FD6618               ld h,(iy+lremb)         ;get remote's address
2058 5E59  FD6E19               ld l,(iy+lreml)
2059 5E5C  E3                   ex (sp),hl              ;change destination
2060 5E5D  C3185E               jp sitlop


                        ;Set Flow

2065 5E60  CDA259       sitflo: call dispth
2066 5E63  50                   defb 'P'
2067 5E64  925E785E             defw sfp,sfptxt
2068 5E68  47                   defb 'G'
2069 5E69  9F5E7F5E             defw sfg,sfgtxt
2070 5E6D  42                   defb 'B'
2071 5E6E  BE5E875E             defw sfb,sfbtxt
2072 5E72  4E                   defb 'N'
2073 5E73  C55E8B5E             defw sfn,sfntxt
```

```
2074 5E77  FF                  defb -1

2076 5E78  726F6365    sfptxt: defm 'rocess',0
2076 5E7C  737300
2077 5E7F  656E6572    sfgtxt: defm 'enerate',0
2077 5E83  61746500
2078 5E87  6F746800    sfbtxt: defm 'oth',0
2079 5E8B  65697468    sfntxt: defm 'either',0
2079 5E8F  657200

2081 5E92  CD585F      sfp:    call sitrsb
2082 5E95  3E01        xsfp:   ld a,1
2083 5E97  0614        xnsg:   ld b,mwpx              ;set process flow for xo
2084 5E99  CDAC5E              call xdouns
2085 5E9C  AF                  xor a                  ;clear generate flow fo
2086 5E9D  180B                jr xsg

2088 5E9F  CD585F      sfg:    call sitrsb
2089 5EA2  AF          xsfg:   xor a
2090 5EA3  0614        xysg:   ld b,mwpx
2091 5EA5  CDAC5E              call xdouns
2092 5EA8  3E01                ld a,1
2093 5EAA  0616        xsg:    ld b,mwgx
2094 5EAC  FDCB005E    xdouns: bit onbit,(iy+ltype1)  ;line must be on!
2095 5EB0  CAD05C              jp z,ccret             ;return without carry (f
2096 5EB3  CDE869              call mpcdo
2097 5EB6  C403FC              call nz,rdie           ; ++ XDOUNS: couldn't d
2098 5EB9  CDA369              call mpcuns
2099 5EBC  37                  scf
2100 5EBD  C9                  ret

2102 5EBE  CD585F      sfb:    call sitrsb
2103 5EC1  3E01        xsfb:   ld a,1
2104 5EC3  18DE                jr xysg

2106 5EC5  CD585F      sfn:    call sitrsb
2107 5EC8  AF          xsfn:   xor a
2108 5EC9  18CC                jr xnsg


                       ;Set Interrupt

2113 5ECB  CDA259      sitint: call dispth
2114 5ECE  42                  defb 'B'
2115 5ECF  075FDE5E            defw sib,sibtxt
2116 5ED3  43                  defb 'C'
2117 5ED4  195FE35E            defw sic,sictxt
2118 5ED8  4E                  defb 'N'
2119 5ED9  385F025F            defw siv,sivtxt
2120 5EDD  FF                  defb -1

2122 5EDE  7265616B    sibtxt: defm 'reak',0
2122 5EE2  00
2123 5EE3  68617261    sictxt: defm 'haracter (enter) ',0
2123 5EE7  63746572
2123 5EEB  2028656E
2123 5EEF  74657229
2123 5EF3  2000
```

```
2124 5EF5 20287265    si2txt: defm ' (re-enter) ',0
2124 5EF9 2D656E74
2124 5EFD 65722920
2124 5F01 00
2125 5F02 65766572    sivtxt: defm 'ever',0
2125 5F06 00

2127 5F07 CD585F      sib:    call sitrsb
2128 5F0A FDCB0086    xsib:   res pbkbit,(iy+ltype1)  ;clear pass break
2129 5F0E AF                  xor a                   ;set interrupt char to z
2130 5F0F 061E        sindo:  ld b,mwint
2131 5F11 CDE869              call mpcdo
2132 5F14 CDA369              call mpcuns
2133 5F17 37                  scf
2134 5F18 C9                  ret

2136 5F19 CD925F      sic:    call imgin              ;get an image-mode chara
2137 5F1C B7                  or a                    ;make sure it's not nul
2138 5F1D CA5F59              jp z,error
2139 5F20 F5                  push af                 ;save it
2140 5F21 21F55E              ld hl,si2txt            ;ask again
2141 5F24 CD9362              call strout
2142 5F27 CD925F              call imgin              ;get another copy
2143 5F2A D1                  pop de                  ;get old one into D
2144 5F2B BA                  cp d                    ;they match?
2145 5F2C C25F59              jp nz,error             ;nope, punt
2146 5F2F CD585F              call sitrsb             ;call remote, A=charact
2147 5F32 FDCB00F6    xsic:   set pbkbit,(iy+ltype1)  ;pass breaks
2148 5F36 18D7                jr sindo

2150 5F38 CD585F      siv:    call sitrsb
2151 5F3B AF          xsiv:   xor a                   ;start with no attn char
2152 5F3C 18F4                jr xsic                 ;merge


                      ;Set Echo

2157 5F3E CDDE62      siteco: call yesno
2158 5F41 2808                jr z,echr               ;is no, echo remote
2159 5F43 CD585F      echl:   call sitrsb
2160 5F46 3E01        xechl:  ld a,1                  ;local echo desired
2161 5F48 FDCB15FE            set eclbit,(iy+lflag2)
2162 5F4C 1808                jr xech

2164 5F4E CD585F      echr:   call sitrsb             ;call confirm, put PC t
2165 5F51 AF          xechr:  xor a
2166 5F52 FDCB15BE            res eclbit,(iy+lflag2)
2167 5F56 0610        xech:   ld b,mwecho
2168 5F58 C3AC5E              jp xdouns
2169 5F5B
```

```
                        ;All the various SET commands come here.  They have args
                        ;remote subroutine is next in-line after the CALL SITRS
                        ;SITRSB never returns, of course, just takes advantage o

2174 5F5B  F5           sitrsb: push af                 ;save AF across CNFIRM
2175 5F5C  CDC562               call cnfirm
2176 5F5F  F1                   pop af                  ;get AF back

2178 5F60  D9                   exx                     ;' point to alternate
2179 5F61  D1                   pop de                  ;' get subr to call
2180 5F62  C1                   pop bc                  ;' and destination.
2181 5F63  C5                   push bc                 ;' restore node.line
2182 5F64  D5                   push de                 ;' and subr address.
2183 5F65  D9                   exx
2184 5F66  CD0268               call rcall              ;call it
2185 5F69  E2B058               jp po,netgon            ;if fails, tell user
2186 5F6C  D25F59               jp nc,error             ;if remote fail, HL has
2187 5F6F  C36E59               jp next                 ;else, all's well
2188 5F72
```

```
                        ;XCL commands

2191 5F72  CD2863       xcl:    call pvchek
2192 5F75  C3DD68               jp xclhi                ;jump to hi memory porti

                        ;Should not be a V-link if it's doing commands.

2196 5F78  FDCB0066     cmdchk: bit vbit,(iy+ltype1)    ;is this a V-link?
2197 5F7C  C8                   ret z
2198 5F7D  CD03FC               call rdie               ;nono...

2200 5F80
```

```
                        ;CHRIN gets a character (using MPCIC) and strips off
                        ;its parity bit.  It also will kill the line if TDOWNP.
                        ;IMGIN leave character alone.

2205 5F80  CD4755       imginw: call dbr                ;enter here to wait a w
2206 5F83  CDF764               call tdownp             ;line down?
2207 5F86  2850                 jr z,idled              ;if so - deadify the lin
2208 5F88  FDCB147E             bit cnvbit,(iy+1flag1)  ;if conversing
2209 5F8C  C40559               call nz,remup           ; is the line up?
2210 5F8F  CD7858               call gontst             ;test for carrier drop.

2212 5F92  CD876A       imgin:  call mpcic
2213 5F95  30E9                 jr nc,imginw            ;if no chars yet, wait
2214 5F97  C9                   ret

2216 5F98  CD925F       chrin:  call imgin
2217 5F9B  E67F                 and low7                ;no parity, please
2218 5F9D  C9                   ret

                        ;BLANK writes a space

2222 5F9E  3E20         blank:  ld a,' '

                        ;CHROUT outputs a character from A using MPCOUT.

2226 5FA0  F5           chrout: push af                 ;save char in case no w
2227 5FA1  CDA66A               call mpcoc
2228 5FA4  300C                 jr nc,chrow             ;if MPCOC failed, wait
2229 5FA6  F1                   pop af                  ;else get chr back
2230 5FA7  C5                   push bc
2231 5FA8  01B400               ld bc,cmdtim            ;set up the command time
2232 5FAB  CDFE64               call settd
2233 5FAE  C1                   pop bc
2234 5FAF  C33E55               jp db                   ;dismiss and return.

2236 5FB2  CDF764       chrow:  call tdownp             ;is the line to be kille
2237 5FB5  CAF05F               jp z,todead             ;if so, kill it (YANK wi
2238 5FB8  F1                   pop af                  ;get chr back
2239 5FB9  CD4755               call dbr                ;wait a while
2240 5FBC  18E2                 jr chrout               ;try again

                        ;CLRBFI eats up all the characters on the keyboard of an

2244 5FBE  CD4755       clrbiw: call dbr
2245 5FC1  0609         clrbfi: ld b,mcli               ;load command code
2246 5FC3  CDE869               call mpcdo
2247 5FC6  C403FC               call nz,rdie
2248 5FC9  060C                 ld b,mr3                ;read KCNT
2249 5FCB  CDE869               call mpcdo
2250 5FCE  3A022C               ld a,(mv1)              ;load keyboard count.
2251 5FD1  B7                   or a                    ;set condition code.
2252 5FD2  CDA369               call mpcuns             ;deselect the MPC.
2253 5FD5  20E7                 jr nz,clrbiw            ;if still left, wait fo
2254 5FD7  C9                   ret                     ;all done, return

2256 5FD8  CD8864       idled:  call todedp
```

```
2258 5FDB  0D0A        idltxt: defb ctrlm,ctrlj
2259 5FDD  203F3031            defm ' ?012 Idle timeout',0
2259 5FE1  32204964
2259 5FE5  6C652074
2259 5FE9  696D656F
2259 5FED  757400

2261 5FF0  118056      todead: ld de,deads
2262 5FF3  CD8D64              call yankme
2263 5FF6
```

```
                        ;ECHIN gets a character using CHRIN, echoes it as approp
                        ;characters, etc.) and converts it to upper case before
                        ;to the caller.

2268 5FF6  CD0260       echin:  call echinl             ;get character (no case
2270 5FF9  FE61                 cp 'a'                  ;maybe convert it to UC
2271 5FFB  D8                   ret c                   ;if .lt. 'a', no need
2272 5FFC  FE7B                 cp 'z'+1
2273 5FFE  D0                   ret nc                  ;or .ge. 'z'+1
2274 5FFF  C6E0                 add a,'A'-'a'           ;make UC
2275 6001  C9                   ret

2277 6002  CD985F       echinl: call chrin              ;get it
2279 6005  FE0D                 cp ctrlm
2280 6007  C8                   ret z
2281 6008  FE7F                 cp 127                  ;including rubout
2282 600A  C8                   ret z

2284 600B  F5                   push af                 ;else type the character
2285 600C  FDCB157E             bit eclbit,(iy+1flag2)  ; (if not already MPC ec
2286 6010  CCA05F               call z,chrout           ;  in the correct pari
2287 6013  F1                   pop af                  ;   then restore it.
2288 6014  C9                   ret
2289 6015
```

```
                        ;Subroutine to read a decimal number into C, delimiter i
                        ;set if got something. Uses AFBC.

2293 6015  CD426C       decinc: call decin              ;enter here if input is
2294 6018  D8                   ret c                   ;okay
2295 6019  FE3F                 cp '?'
2296 601B  200D                 jr nz,decine
2297 601D  CD5E59               call errorp
2298 6020  2041206E             defm ' A number',0
2298 6024  756D6265
2298 6028  7200

2300 602A  CD5E59       decine: call errorp
2301 602D  203F3031             defm ' ?013 Illegal number',0
2301 6031  3320496C
2301 6035  6C656761
2301 6039  6C206E75
2301 603D  6D626572
2301 6041  00

2303 6042  010000       decin:  ld bc,0                 ;start fresh
2304 6045  D5                   push de                 ;save a temp

2306 6046  CDF65F       decin1: call echin              ;get a character into A
2307 6049  FE30                 cp '0'                  ;in range?
2308 604B  3817                 jr c,decin3             ;if .lt. '0', illegal
2309 604D  FE3A                 cp '9'+1
2310 604F  3013                 jr nc,decin3            ;if .ge. '9' + 1, illega

2312 6051  D630                 sub '0'                 ;make A into a number
2313 6053  5F                   ld e,a                  ;save it
2314 6054  79                   ld a,c                  ;get accumulated digits
2315 6055  87                   add a,a                 ;times two
2316 6056  87                   add a,a                 ;times 4
2317 6057  87                   add a,a                 ;times 8
2318 6058  81                   add a,c                 ;times 9
2319 6059  81                   add a,c                 ;times 10
2320 605A  83                   add a,e                 ;plus new digit

2322 605B  B9                   cp c                    ;compare with old value
2323 605C  DA2A60               jp c,decine             ;if new .lt. old, overf
2324 605F  4F                   ld c,a                  ;put it back
2325 6060  CBC0                 set 0,b                 ;mark having got one cha

2327 6062  18E2                 jr decin1               ;loop for more

2329 6064  D1           decin3: pop de                  ;restore de
2330 6065  37                   scf                     ;assume got something
2331 6066  CB40                 bit 0,b                 ;get anything?
2332 6068  C0                   ret nz                  ;yes, return
2333 6069  B7                   or a                    ;(doesn't destroy A)
2334 606A  C9                   ret                     ;assumtion wrong, flip c
2335 606B
```

```
                        ;DECOUT types C in decimal. Uses AF.

2338 6068  C5           decout: push bc                 ;save BC.
2339 606C  E5                   push hl                 ;save HL.
2340 606D  79                   ld a,c                  ;obtain argument.
2341 606E  CD746C               call decdo              ;call recursive decimal
2342 6071  E1                   pop hl                  ;restore HL.
2343 6072  C1                   pop bc                  ;restore BC.
2344 6073  C9                   ret

2346 6074  06FF         decdo:  ld b,0ffh               ;pre-decrement quotien

2348 6076  04           decott: inc b                   ;add one to quotient
2349 6077  D60A                 sub 10                  ;subtract 10.
2350 6079  30FB                 jr nc,decott            ;and try again.

2352 607B  C60A                 add a,10                ;reconstitute remainder
2353 607D  F5                   push af                 ;save remainder.
2354 607E  78                   ld a,b                  ;check quotient to see i
2355 607F  B7                   or a                    ; ...
2356 6080  C4746C               call nz,decdo           ;wasn't zero, must recur
2357 6083  F1                   pop af                  ;restore remainder.
2358 6084  C630                 add a,'0'               ;convert to ASCII.
2359 6086  C3A05F               jp chrout

                        ;DEC3R outputs C as a 3-digit decimal, right justified
                        ;DEC3L is same but left-justified.

2364 6089  CD996C       dec3r:  call decpad             ;pad the number
2365 608C  CD6B6C               call decout             ;output it
2366 608F  C9                   ret                     ;return

2368 6090  C5           dec3l:  push bc                 ;save a copy
2369 6091  CD6B6C               call decout             ;output it
2370 6094  C1                   pop bc
2371 6095  CD996C               call decpad             ;pad it
2372 6098  C9                   ret

                        ;DECPAD outputs the right number of spaces for the numbe

2376 6099  79           decpad: ld a,c                  ;get number in questio
2377 609A  FE64                 cp 100                  ;3 digit?
2378 609C  D0                   ret nc                  ;yes, no padding
2379 609D  FE0A                 cp 10                   ;2 digit?
2380 609F  DC9E5F               call c,blank            ;if 1 digit, output 2 sp
2381 60A2  CD9E5F               call blank              ;else output 1 space
2382 60A5  C9                   ret
2383 60A6
```

```
                                ;RATIN and RATOUT read an write baud rates from A (not C
2386 60A6  E5           ratin:  push hl                 ;save HL and BC
2387 60A7  C5                   push bc
2388 60A8  CD156C               call decinc             ;get the baud rate
2389 60AB  FE3F                 cp '?'                  ;want help?
2390 60AD  2828                 jr z,rtihlp
2391 60AF  FE68                 cp 'h'                  ;or HUNT?
2392 60B1  2818                 jr z,rtihnt
2393 60B3  CDF662               call whchek             ;needs whitespace afterw
2394 60B6  79                   ld a,c                  ;move number to A
2395 60B7  21F66C               ld hl,rtitab            ;get pointer to rate tab
2396 60BA  0610                 ld b,16                 ;there are 16 rates
2397 60BC  BE           rtilop: cp (hl)                 ;a match?
2398 60BD  2806                 jr z,rtimat
2399 60BF  23                   inc hl                  ;no, step to next
2400 60C0  10FA                 djnz rtilop             ;loop
2401 60C2  C32A6C       rtierr: jp decine               ;bad number

2403 60C5  3E10         rtimat: ld a,16                 ;compute 16-B = entry r
2404 60C7  90                   sub b
2405 60C8  C1                   pop bc
2406 60C9  E1                   pop hl
2407 60CA  C9                   ret                     ;and return

2409 60CB  CD8A62       rtihnt: call stroth             ;hunt
2410 60CE  756E7400             defm 'unt',0
2411 60D2  3EFF                 ld a,0FFh               ;get FF = hunt code
2412 60D4  C1                   pop bc
2413 60D5  E1                   pop hl
2414 60D6  C9                   ret

2416 60D7  CD5E59       rtihlp: call errorp             ;he wants help
2417 60DA  46697273             defm 'First 2 digits of baud rate',0
2417 60DE  74203220
2417 60E2  64696769
2417 60E6  7473206F
2417 60EA  66206261
2417 60EE  75642072
2417 60F2  61746500

2419 60F6  324B080D     rtitab: defb 50,75,11,13
2420 60FA  0F1E3C0C             defb 15,30,60,12
2421 60FE  12141824             defb 18,20,24,36
2422 6102  30486013             defb 48,72,96,19

2424 6106  E5           ratout: push hl
2425 6107  C5                   push bc
2426 6108  FEFF                 cp 0FFh                 ;hunt?
2427 610A  212161                ld hl,rtohnt            ;in case hunt
2428 610D  280C                 jr z,rtoout
2429 610F  87                   add a,a                 ;compute 2 * rate
2430 6110  4F                   ld c,a                  ;save a copy
2431 6111  87                   add a,a                 ;4 *
2432 6112  87                   add a,a                 ;8 *
2433 6113  91                   sub c                   ;8 * - 2 * = 6 *
2434 6114  4F                   ld c,a
```

```
2435 6115  0600              ld b,0              ;get BC = 6 * rate
2436 6117  212761            ld hl,rtotab
2437 611A  09                add hl,bc
2438 611B  CD9362    rtoout: call strout         ;output it
2439 611E  C1                pop bc
2440 611F  E1                pop hl
2441 6120  C9                ret

2443 6121  48756E74  rtohnt: defm 'Hunt ',0
2443 6125  2000
2444 6127  35302020  rtotab: defm '50   ',0
2444 612B  2000
2445 612D  37352020          defm '75   ',0
2445 6131  2000
2446 6133  31313020          defm '110  ',0
2446 6137  2000
2447 6139  3133342E          defm '134.5',0
2447 613D  3500
2448 613F  31353020          defm '150  ',0
2448 6143  2000
2449 6145  33303020          defm '300  ',0
2449 6149  2000
2450 614B  36303020          defm '600  ',0
2450 614F  2000
2451 6151  31323030          defm '1200 ',0
2451 6155  2000
2452 6157  31383030          defm '1800 ',0
2452 615B  2000
2453 615D  32303030          defm '2000 ',0
2453 6161  2000
2454 6163  32343030          defm '2400 ',0
2454 6167  2000
2455 6169  33363030          defm '3600 ',0
2455 616D  2000
2456 616F  34383030          defm '4800 ',0
2456 6173  2000
2457 6175  37323030          defm '7200 ',0
2457 6179  2000
2458 617B  39363030          defm '9600 ',0
2458 617F  2000
2459 6181  31393230          defm '19200',0
2459 6185  3000
2460 6187
```

```
                        ;GRPIN and GRPOUT read and write group specifications th

2463 6187  D5           grpin:  push de
2464 6188  C5                   push bc
2465 6189  1E00                 ld e,0                  ;clear accumulated grou
2466 618B  CDF65F       gpilop: call echin              ;get another character
2467 618E  FE3F                 cp '?'
2468 6190  281F                 jr z,gpihlp             ;give help
2469 6192  FE20                 cp ' '                  ;done?
2470 6194  2813                 jr z,gpidon
2471 6196  FE38                 cp '8'
2472 6198  DA2A60               jp c,decine             ;less than 0
2473 619B  D630                 sub '0'                 ;convert to 0 to 7
2474 619D  47                   ld b,a                  ;get count (1-7)
2475 619E  3E01                 ld a,1                  ;start with 1
2476 61A0  2803                 jr z,gpizro             ;if count is zero alread
2477 61A2  87           gpiadd: add a,a                 ;shift left
2478 61A3  10FD                 djnz gpiadd             ;until done
2479 61A5  B3           gpizro: or e                    ;update E
2480 61A6  5F                   ld e,a
2481 61A7  18E2                 jr gpilop               ;and accumulated more

2483 61A9  7B           gpidon: ld a,e                  ;copy accumulated grou
2484 61AA  B7                   or a                    ;make sure he typed some
2485 61AB  CA2A60               jp z,decine             ;he didn't
2486 61AE  C1                   pop bc
2487 61AF  D1                   pop de
2488 61B0  C9                   ret

2490 61B1  CD5E59       gpihlp: call errorp
2491 61B4  0D0A                 defb ctrlm,ctrlj
2492 61B6  456E7465             defm 'Enter a group specification',0
2492 61BA  72206120
2492 61BE  67726F75
2492 61C2  70207370
2492 61C6  65636966
2492 61CA  69636174
2492 61CE  696F6E00

2494 61D2  D5           grpout: push de                 ;save some regs
2495 61D3  C5                   push bc
2496 61D4  113001               ld de,(1*256) + '0'     ;D=01, E="0" for group 0
2497 61D7  0608                 ld b,8                  ;and there are eight of
2498 61D9  F5           gpolop: push af                 ;save group word
2499 61DA  A2                   and d                   ;this one on?
2500 61DB  3E2D                 ld a,'-'                ;assume not - dash
2501 61DD  2801                 jr z,gpoout
2502 61DF  7B                   ld a,e                  ;yes - output correspond
2503 61E0  CDA05F       gpoout: call chrout
2504 61E3  CB22                 sla d                   ;shift D left
2505 61E5  1C                   inc e                   ;next character
2506 61E6  F1                   pop af
2507 61E7  10F0                 djnz gpolop             ;until have done them a
2508 61E9  C1                   pop bc
2509 61EA  D1                   pop de
2510 61EB  C9                   ret
2511 61EC
```

```
                        ;DSTIN accepts destinations of the form <node_number>.<l
                        ;<host_number>.  Destination is returned in BC, trash i
                        ;FFhh, second nnll.  Uses AFBC.  Goes to ERROR if bad de
                        ;terminated by space or carriage return.

2517 61EC CD156C        dstin:  call decinc             ;read decimal number.
2518 61EF FE2E                  cp '.'                  ;a two-part dest?
2519 61F1 2806                  jr z,dstin2
2520 61F3 CDF662                call whchek             ;no, better be whitespac
2521 61F6 06FF                  ld b,0ffh               ;make FFhh where hh=host
2522 61F8 C9                    ret

2524 61F9 CDA863        dstin2: call bxchek             ;check C for legal node
2525 61FC D5                    push de                 ;save DE
2526 61FD 51                    ld d,c                  ;copy first number to B
2527 61FE CD156C                call decinc             ;get second part
2528 6201 CDF662                call whchek             ;check for "white spac
2529 6204 CD8A63                call lnchek             ;legal line number?
2530 6207 42                    ld b,d                  ;combine node number wit
2531 6208 D1                    pop de                  ;restore DE
2532 6209 C9                    ret

                        ;LNKIN is just like DSTIN except host address are not a

2536 620A E5            lnkin:  push hl                 ;save hl
2537 620B CD156C                call decinc             ;get the node part
2538 620E 212462                ld hl,lkitxt            ;get help text
2539 6211 FE3F                  cp '?'                  ;help?
2540 6213 CA5F59                jp z,error              ;yes, give help
2541 6216 FE2E                  cp '.'
2542 6218 C22A6C                jp nz,decine            ;must end in dot
2543 621B 61                    ld h,c                  ;get high part of addr
2544 621C CD156C                call decinc             ;get low part
2545 621F CDF662                call whchek
2546 6222 44                    ld b,h                  ;get high parg back int
2547 6223 C9                    ret                     ;done

2549 6224 0D0A          lkitxt: defb ctrlm,ctrlj
2550 6226 53706563              defm 'Specify physical network address',0
2550 622A 69667920
2550 622E 70687973
2550 6232 6963616C
2550 6236 206E6574
2550 623A 776F726B
2550 623E 20616464
2550 6242 72657373
2550 6246 00
2551 6247
```

```
                        ;DSTOUT, subroutine to output a destination from BC. Use

2554 6247  78           dstout: ld a,b                  ;which type?
2555 6248  3C                   inc a
2556 6249  2004                 jr nz,dsto2             ;if b<>FF, two part
2557 624B  CD6B6C               call decout             ;just output C
2558 624E  C9                   ret

2560 624F  C5           dsto2:  push bc                 ;save it
2561 6250  48                   ld c,b                  ;get hi byte
2562 6251  CD6B6C               call decout
2563 6254  3E2E                 ld a,'.'                ;separate with dot
2564 6256  CDA05F               call chrout
2565 6259  C1                   pop bc                  ;get back low part
2566 625A  CD6B6C               call decout
2567 625D  C9                   ret

                        ;DSTFOU outputs a formatted destination.  Formatted dest
                        ;take up 5 characters, either nnn.nnn or H nnn where num
                        ;padded with blanks as appropriate.  Uses AFBC.

2573 625E  C5           dstfou: push bc                 ;save caller's arg
2574 625F  48                   ld c,b                  ;which type?
2575 6260  04                   inc b                   ;0FF?
2576 6261  280D                 jr z,dstfoh             ;yes, host
2577 6263  CD896C               call dec3r              ;output node address fr
2578 6266  3E2E                 ld a,'.'
2579 6268  CDA05F               call chrout
2580 626B  C1                   pop bc                  ;get line address
2581 626C  CD906C               call dec31              ;left justified
2582 626F  C9                   ret

2584 6270  3E48         dstfoh: ld a,'H'                ;get host
2585 6272  CDA05F               call chrout
2586 6275  CD9E5F               call blank              ;and a space
2587 6278  C1                   pop bc                  ;get host number
2588 6279  CD906C               call dec31
2589 627C  C9                   ret
2590 627D
```

```
                        ;BOXIN reads a node number and forms <nodenum>.0 in BC.

2593 627D  CD156D       boxin:  call decinc             ;get a character
2594 6280  CDF662               call whchek
2595 6283  CDA863               call bxchek             ;better be okay
2596 6286  41                   ld b,c                  ;move to B
2597 6287  0E00                 ld c,0                  ;form node.0 (reasonable
2598 6289  C9                   ret
2599 628A
```

```
                        ;STROUT types string in HL.  Uses AFHL.  Enter at CRLF t
                        ;On return, HL points to the terminating zero byte.

2603 628A  E1           stroth: pop hl                  ;enter here to type a st
2604 628B  CD9362               call strout             ;type it
2605 628E  23                   inc hl                  ;skip over the terminati
2606 628F  E9                   jp (hl)                 ;and return to caller

2608 6290  21E159       crlf:   ld hl,crltxt            ;feed it to STROUT
2609 6293  7E           strout: ld a,(hl)               ;was this the string end
2610 6294  B7                   or a
2611 6295  C8                   ret z                   ;yup, return. else ...

2613 6296  CD036B               call mpcptl             ;get printer character
2614 6299  B7                   or a                    ;any?
2615 629A  281D                 jr z,strowt             ;no, wait for at least o

2617 629C  D5                   push de                 ;save DE
2618 629D  C5                   push bc                 ;save BC
2619 629E  47                   ld b,a                  ;copy max string size t
2620 629F  E5                   push hl                 ;save string start.
2621 62A0  C5                   push bc                 ;save starting count.
2622 62A1  AF                   xor a                   ;get a nul
2623 62A2  BE           stroul: cp (hl)                 ;is this the string end?
2624 62A3  2803                 jr z,strous             ;yup, send it.
2625 62A5  23                   inc hl                  ;get to next character.
2626 62A6  10FA                 djnz stroul             ;loop if there's space.
2627 62A8  78           strous: ld a,b                  ;get ending count.
2628 62A9  C1                   pop bc                  ;restore starting count
2629 62AA  90                   sub b                   ;get difference.
2630 62AB  ED44                 neg                     ;make positive...will ty
2631 62AD  54                   ld d,h                  ;start next string
2632 62AE  5D                   ld e,l                  ; at this location.
2633 62AF  E1                   pop hl                  ;restore string start.
2634 62B0  0603                 ld b,mwt                ;write text
2635 62B2  CD4A6A               call mpcout             ; to MPC.
2636 62B5  62                   ld h,d                  ;reload string continuat
2637 62B6  6B                   ld l,e                  ; address.
2638 62B7  C1                   pop bc                  ;restore saved ACs.
2639 62B8  D1                   pop de                  ; ...

2641 62B9  CD4755       strowt: call dbr                ;wait for the buffer to
2642 62BC  18D5                 jr strout               ; empty, then try again.
2643 62BE
```

```
                        ;CNFRM checks for already input carriage return.  If non
                        ;wait for one.

2647 62BE  FE0D         cnfrm:  cp ctrlm                ;a CR already in?
2648 62C0  C8                   ret z                   ;yup, return right away
2649 62C1  FE20                 cp ' '                  ;must be a space
2650 62C3  2037                 jr nz,noncnf            ; or give error.

                        ;CNFIRM prints [Confirm], waits for a CR, else goes to e

2654 62C5  21D262       cnfirm: ld hl,cnftxt            ;get message
2655 62C8  CD9362               call strout
2656 62CB  CDF65F               call echin              ;get char
2657 62CE  CDF962               call crchek             ;must be CR
2658 62D1  C9                   ret

2660 62D2  205B636F     cnftxt: defm ' [confirm] ',0
2660 62D6  6E666972
2660 62DA  6D5D2C00

                        ;YESNO gets a YES or NO response, Z=no, NZ=yes

2664 62DE  CDA259       yesno:  call dispth
2665 62E1  59                   defb 'Y'
2666 62E2  F162EC62             defw yes,yestxt
2667 62E6  4E                   defb 'N'
2668 62E7  F462EF62             defw no,notxt
2669 62EB  FF                   defb -1

2671 62EC  657300       yestxt: defm 'es',0
2672 62EF  6F00         notxt:  defb 'o',0

2674 62F1  F601         yes:    or 1                    ;set the NZ bit
2675 62F3  C9                   ret

2677 62F4  AF           no:     xor a                   ;clear the NZ bit
2678 62F5  C9                   ret

                        ;WHCHEK checks A agains CR or space. See CRCHEK.

2682 62F6  FE20         whchek: cp ' '                  ;a space?
2683 62F8  C8                   ret z                   ;return if yes, fall int

                        ;CRCHEK checks character in A against carriage return.
                        ;illegal confirmation and goes to ERROR.

2688 62F9  FE0D         crchek: cp ctrlm                ;a carriage return?
2689 62FB  C8                   ret z
2690 62FC  CD5E59       noncnf: call errorp
2691 62FF  203F3031             defm ' ?014 Not confirmed',0
2691 6303  34204E6F
2691 6307  7420636F
2691 630B  6E666972
2691 630F  6D656400

                        ;ABORT checks to see whether the character in A is any
                        ;characters.  If so, go to ERROR, else return with A unc
```

```
2696 6313  FE7F        abort:  cp rubout
2697 6315  2807                jr z,abtabt
2698 6317  FE08                cp ctrlh
2699 6319  2803                jr z,abtabt
2700 631B  FE18                cp ctrlx
2701 631D  C0                  ret nz
2702 631E  CD5E59      abtabt: call errorp
2703 6321  20204162            defm '  Aborted',0
2703 6325  6F727465
2703 6329  6400
2704 632B
```

```
                        ;PVCHEK checks whether this user is prived

 2707 6328  FDCB077E    pvchek: bit g7bit,(iy+logrp)    ;is he in group 7?
 2708 632F  C0                   ret nz                  ;yes
 2709 6330  CD5E59               call errorp
 2710 6333  203F3030             defm ' ?000 Must be administrator',0
 2710 6337  30204D75
 2710 633B  73742062
 2710 633F  65206164
 2710 6343  6D696E69
 2710 6347  73747261
 2710 634B  746F7200

                        ;CNCHEK checks whether this user is in a conversation

 2714 634F  FDCB147E    cnchek: bit cnvbit,(iy+1flag1)
 2715 6353  C0                   ret nz                  ;if yes, fine
 2716 6354  CD5E59               call errorp
 2717 6357  203F3031             defm ' ?015 Not connected',0
 2717 635B  35204E6F
 2717 635F  7420636F
 2717 6363  6E6E6563
 2717 6367  74656400

                        ;DSCCHK checks whether this user is in a conversation

 2721 636B  FDCB147E    dscchk: bit cnvbit,(iy+1flag1)
 2722 636F  C8                   ret z                   ;if not, fine
 2723 6370  CD5E59               call errorp
 2724 6373  203F3031             defm ' ?019 Disconnect first',0
 2724 6377  39204469
 2724 637B  73636F6E
 2724 637F  6E656374
 2724 6383  20666972
 2724 6387  737400

                        ;LNCHEK checks whether number in C is a legal link numbe

 2728 638A  F5           lnchek: push af
 2729 638B  79                   ld a,c
 2730 638C  FE20                 cp nlink
 2731 638E  3816                 jr c,rets
 2732 6390  CD5E59               call errorp
 2733 6393  203F3031             defm ' ?016 Illegal link',0
 2733 6397  3620496C
 2733 639B  6C656761
 2733 639F  6C206C69
 2733 63A3  6E6B00

 2735 63A6  F1           rets:   pop af
 2736 63A7  C9                   ret

                        ;BXCHEK checks whether number in C (not A!) is a legal
                        ;Goes to error if not. Uses AF.

 2741 63A8  F5           bxchek: push af
 2742 63A9  79                   ld a,c                  ;get it
```

```
2743 63AA  FE10          cp nbox                ;okay?
2744 63AC  38F8          jr c,rets              ;if .lt. nbox, fine (box
2745 63AE  CD5E59        call errorp
2746 63B1  203F3031      defm ' ?017 Illegal node',0
2746 63B5  3720496C
2746 63B9  6C656761
2746 63BD  6C206E6F
2746 63C1  646500
2747 63C4
```

```
                        ;Subroutines to GET and GIV the XCL command lock.  These
                        ;here (instead of in D.Z80) because DEAD and NEXT call

2751 63C4  3A8842       getlok: ld a,(xcllok)           ;get the lock
2752 63C7  FEFF                 cp 0FFh                 ;empty?
2753 63C9  2840                 jr z,gotlok
2754 63CB  F5                   push af                 ;save lock
2755 63CC  CD8A62               call stroth             ;tell user who is using
2756 63CF  203F3C31             defm ' ?010 Command interlocked by link ',0
2756 63D3  3C20436F
2756 63D7  6D6D616E
2756 63DB  6420696E
2756 63DF  7465726C
2756 63E3  6F636865
2756 63E7  64206279
2756 63EB  206C696E
2756 63EF  682000
2757 63F2  F1                   pop af                  ;put it into BC
2758 63F3  4F                   ld c,a
2759 63F4  3A0042               ld a,(me)
2760 63F7  47                   ld b,a
2761 63F8  CD4762               call dstout
2762 63FB  CD5E59               call errorp
2763 63FE  2C202074             defm ' - try later',0
2763 6402  7279206C
2763 6406  61746572
2763 640A  00

2765 640B  FD7E0D       gotlok: ld a,(iy+lnum)
2766 640E  328842               ld (xcllok),a           ;take lock
2767 6411  C9                   ret                     ;and return

2769 6412  3A8842       givlok: ld a,(xcllok)           ;a match?
2770 6415  FDAE0D               xor (iy+lnum)
2771 6418  C0                   ret nz                  ;no, we don't own the lo
2772 6419  3D                   dec a                   ;get FF
2773 641A  328842               ld (xcllok),a           ;yes, give lock back
2774 641D  C9                   ret
2775 641E
```

```
                        ;Subroutine to disconnect any pending connection.

2778 641E  FDCB147E     discon: bit cnvbit,(iy+1flag1)  ;in a conversation?
2779 6422  C8                   ret z                   ;no - return immediately
2780 6423  CDD559               call remup              ;see if remote is reach
2781 6426  3011                 jr nc,diszap            ;if not, just zap CNV aw

2783 6428  FD5618               ld d,(iy+1remb)         ;yes - RCALL to discon
2784 642B  FD5E19               ld e,(iy+1reml)
2785 642E  D5                   push de
2786 642F  214164               ld hl,xdsc              ;remote disconnect
2787 6432  E5                   push hl
2788 6433  FD7E17               ld a,(iy+1cid)          ;offer sequence
2789 6436  CDD268               call rcall
2790 6439  FDCB14BE     diszap: res cnvbit,(iy+1flag1)  ;ignore errors
2791 643D  FD341A               inc (iy+1watch)         ;for WATCH to see
2792 6440  C9                   ret

                        ;Remote subr: XDSC. Enter with SEQ in A. Checks FRB/FRL
                        ;LREMB/LREML to be sure, too.

2797 6441  FDBE17       xdsc:   cp (iy+1cid)            ;sequence match?
2798 6444  C403FC               call nz,rdie            ; ++ XDSC: cid mismatch
2799 6447  DD7E08               ld a,(ix+pfrb)          ;check sender, too
2800 644A  E67F                 and low7
2801 644C  FDBE18               cp (iy+1remb)
2802 644F  C403FC               call nz,rdie            ; ++ XDSC: source mismat
2803 6452  DD7E09               ld a,(ix+pfrl)
2804 6455  E67F                 and low7
2805 6457  FDBE19               cp (iy+1reml)
2806 645A  C403FC               call nz,rdie            ; ++ XDSC: source mismat
2807 645D  FDCB14BE             res cnvbit,(iy+1flag1)  ;remove from the conver
2808 6461  216C64               ld hl,rdstxt
2809 6464  119B56               ld de,deado             ;kill the line
2810 6467  CDA164               call yankhm             ;put the (our) job in D
2811 646A  37                   scf                     ;give good return
2812 646B  C9                   ret

2814 646C  0D0A         rdstxt: defb ctrlm,ctrlj
2815 646E  203F3130             defm ' ?103 Remote disconnected',0
2815 6472  33205265
2815 6476  6D6F7465
2815 647A  20646973
2815 647E  636F6E6E
2815 6482  65637465
2815 6486  6400
2816 6488
```

```
                        ;TODEDO yanks to DEADO.  TODEDP does also, but string ad

2819 6488  E1           todedp: pop hl                  ;caller's address is als
2820 6489  E5                   push hl                 ; address, drop message
2821 648A  119B56       todedo: ld de,deado             ;off to DEADO
                        ;       jp yankme

                        ;Subroutine to yank from one state to another. Enter wi
                        ;DE=new state, IY=line block. Sets up SP and jumps to ne
                        ;state.  As with YANKHM, must preserve hl

2828 648D  C1           yankme: pop bc
2829 648E  FD712D               ld (iy+llyank+0),c
2830 6491  FD702E               ld (iy+llyank+1),b
2831 6494  FDE5                 push iy                 ;copy IY into IX
2832 6496  DDE1                 pop ix
2833 6498  017F00               ld bc,lstack            ;get stack address
2834 649B  DD09                 add ix,bc               ;hl=address of stack
2835 649D  DDF9                 ld sp,ix                ;set up SP
2836 649F  D5                   push de                 ;put process start addr
2837 64A0  C9                   ret                     ;and go for it.

                        ;Subroutine to put some other line into a new state (in
                        ;tricky because it requires setting up a new stack for t
                        ;Note crafty code to preserve HL to the called routine.

2843 64A1  E3           yankhm: ex (sp),hl
2844 64A2  FD752D               ld (iy+llyank+0),l      ;save pc of last yanker
2845 64A5  FD742E               ld (iy+llyank+1),h      ; the line block.
2846 64A8  E3                   ex (sp),hl
2847 64A9  DDE5                 push ix                 ;save both IX
2848 64AB  C5                   push bc                 ; and BC.
2849 64AC  FDE5                 push iy                 ;compute his stack base
2850 64AE  DDE1                 pop ix                  ; ...
2851 64B0  017F00               ld bc,lstack            ; by adding stack offse
2852 64B3  DD09                 add ix,bc               ; ...
2853 64B5  ED737242             ld (temp1),sp           ;save our stack (probabl
2854 64B9  DDF9                 ld sp,ix                ;move to his
2855 64BB  D5                   push de                 ;save new state as oldes
2856 64BC  F5                   push af
2857 64BD  C5                   push bc
2858 64BE  D5                   push de
2859 64BF  E5                   push hl
2860 64C0  DDE5                 push ix
2861 64C2  210000               ld hl,0
2862 64C5  39                   add hl,sp               ;read his sp
2863 64C6  FD741C               ld (iy+lsp+1),h         ;save it
2864 64C9  FD750F               ld (iy+lsp+0),l
2865 64CC  ED7B7242             ld sp,(temp1)           ;get our stack back
2866 64D0  C1                   pop bc                  ;restore BC
2867 64D1  DDE1                 pop ix                  ; and IX.
2868 64D3  C9                   ret                     ;return
2869 64D4
```

```
                        ;Subroutine to advance IY to the next line block. Return
                        ;at end of lines.

2873 64D4  AF           iyend:  xor a                   ;clear carry - from IYNE
2874 64D5  C9                   ret                     ;return

2876 64D6  FD7E0D       iynext: ld a,(iy+lnum)          ;where are we now?
2877 64D9  FE1F                 cp nlink-1              ;at the last one
2878 64DB  28F7                 jr z,iyend              ;if so, clear carry and
2879 64DD  3C                   inc a                   ;if not, step to next

                        ;Subroutine to transform a line number (in A) into a lin
                        ;block pointer in IY.

2884 64DE  FE20         iylin:  cp nlink                ;make sure it's in range
2885 64E0  D403F0               call nc,rdie            ; ++ IYLIN arg too big
2886 64E3  E5                   push hl
2887 64E4  67                   ld h,a                  ;compute 256*a
2888 64E5  2E00                 ld l,0                  ;in HL
2889 64E7  CB3C                 srl h                   ;divide HL by two
2890 64E9  CB1D                 rr l                    ;yielding 128*a in HL wh
2891 64EB  C5                   push bc                 ;offset by first block
2892 64EC  011B43               ld bc,lblk0
2893 64EF  09                   add hl,bc
2894 64F0  C1                   pop bc                  ;don't trash bc
2895 64F1  E5                   push hl
2896 64F2  FDE1                 pop iy                  ;copy into IY
2897 64F4  E1                   pop hl
2898 64F5  37                   scf                     ;set carry for IYNEXT,
2899 64F6  C9                   ret

                        ;Subroutine TDOWNP returns Z if TDOWN=0, NZ otherwise.

2903 64F7  FD7E12       tdownp: ld a,(iy+ltdown)        ;look at the double-prec
2904 64FA  FDB613               or (iy+ltdown+1)
2905 64FD  C9                   ret

                        ;Subroutine to copy BC into TDOWN.

2909 64FE  FD7013       settd:  ld (iy+ltdown+1),b      ;hi order
2910 6501  FD7112               ld (iy+ltdown+0),c      ;lo order
2911 6504  C9                   ret

                        ;Subroutine to simulate a call (hl) which the Z-80 does

2915 6505  E5           callhl: push hl                 ;call (HL) from pc of ca
2916 6506  C9                   ret
2917 6507
```

```
                        ;Subroutine to test a word of memory (pointed to by HL)
                        ;contents.  It works by floating a one, then a zero th
                        ;float loop always goes "one too far" which will also te
                        ;00000000 and 11111111.  MEMCHK has duplicate code (call
                        ;which may not be put into subroutine form since MEMCHK
                        ;stack at the time.

2925 6507  54           memchk: ld d,h                  ;save HL
2926 6508  5D                   ld e,l
2927 6509  01E29A               ld bc,-memch0           ;make sure address isn't
2928 650C  09                   add hl,bc               ;  MEMCH1
2929 650D  300A                 jr nc,memcok            ;if HL .LT. MEMCH0, okay
2930 650F  01D4FF               ld bc,memch0-memch1     ;now make sure it's not
2931 6512  09                   add hl,bc               ;HL contains some posit
2932 6513                                               ;negative length of the
2933 6513  CB7C                 bit 7,h                 ;check to see if HL has
2934 6515  2802                 jr z,memcok             ;all is ok, proceed
2935 6517  EB                   ex de,hl                ;if so, return since off
2936 6518  C9                   ret                     ; too low. (restore HL f
2937 6519  EB           memcok: ex de,hl                ;fix HL (saved in DE)
2938 651A  5E                   ld e,(hl)               ;save old value
2939 651B  3E01                 ld a,1
2940 651D  0609                 ld b,9
2941 651E                memch0: equ $-1                ;MEMCH0 = first address
2942 651F  3600          mem1lp: ld (hl),0
2943 6521  77                   ld (hl),a
2944 6522  BE                   cp (hl)
2945 6523  C403FC               call nz,rdie            ; ++ MEMONE: error at HL
2946 6526  36FF                 ld (hl),0FFh
2947 6528  77                   ld (hl),a
2948 6529  BE                   cp (hl)
2949 652A  C403FC               call nz,rdie            ; ++ MEMONE: error at HL
2950 652D  CB27                 sla a                   ;try next one
2951 652F  10EE                 djnz mem1lp             ;including the extra tes

2953 6531  3EFE                 ld a,011111110b
2954 6533  0609                 ld b,9
2955 6535  3600          mem0lp: ld (hl),0
2956 6537  77                   ld (hl),a
2957 6538  BE                   cp (hl)
2958 6539  C403FC               call nz,rdie            ; ++ MEMONE: error at HL
2959 653C  36FF                 ld (hl),0FFh
2960 653E  77                   ld (hl),a
2961 653F  BE                   cp (hl)
2962 6540  C403FC               call nz,rdie            ; ++ MEMONE: error at HL
2963 6543  CB27                 sla a
2964 6545  F601                 or 1                    ;for low bit back on
2965 6547  10EC                 djnz mem0lp

2967 6549  73                   ld (hl),e               ;fix word
2968 654A                memch1:                        ;MEMCH1 = first word we
2969 654A  C9                   ret                     ;return to caller
2970 654B
```

```
;This is the routing algorithm driver. This algorithm is
;pre-SPF ARPAnet routing as described by Dr. John McQui
;Algorithms for Computer Networks - A Survey".

;At the heart of the algorithm is the "NETWORK DELAY TA
;
;           destination                          (in this case
;         00  me  02  03  04  ... nbox-1
;         ------------------------------
; l 00!   14  06  07  12  05        21
; i 01!   08  08  07  05  08        18
; n 02!   06  07  06  08  04        19
; e 03!   FF  FE  FF  FF  FE        FE
;
;
;From this table, two important "per-destination" sub-ta
;The first is the "MINIMUM DELAY TABLE" which contains
;in the NDT. Additionally, the "ROUTING TABLE" contains
;axis) associated with columnar MIN:
;
;           destination
;         00  me  02  03  04  ... nbox-1
;         ------------------------------
;MIN DLY  06  00  06  06  04        18
;ROUTE    02  00  02  01  02        01
;
;Every "RUPTIM" time units, each node sends its MIN DLY
;neighbors. When one receives a MIN DLY table, one uses
;NDT row corresponding to the trunk line the table came
;process is as follows:
;
; 0. Increase the incoming message's entries by the cos
;    the link the update came in on.
; 1. Compare each entry in the incoming message with tha
;    overwrite the row's entry and dispatch on the resu
;      NEW < OLD: update the MIN DLY entry to = NEW and
;      NEW = OLD: do nothing
;      NEW > OLD: if the old RUT was out this line, fin
;                 and RUT because another path may now b
;
;In VTN, the delay of a path is the accumulation of 1/r
;where "rn" is the baud rate of hop #n. In VTN, baud rat
;19.2k baud, numbered from 0 to F hexidecimal. We index
;"CTAB" to come up with the delay associated with a lin
;19.2 is assigned a delay of "1", 9600 "2", 4800 "4", et
;with slow hop rates, this sum may exceed 8 bits. In the
;value "overflow" (FE hex) is placed in the NDT for tha
;never gets any larger.
;
;In addition, the value "FF" means un-reachable. When "C
;tables are missed in a row from a neighbor, a dummy MIN
;made up which has all "unreachable" entries. This will
;a re-computation of all paths which used to go out tow
```

```
                        ;Call RUP with IY pointing to line which gets updated in
                        ;HL pointing to beginning of vector (size NBOX) contain

3028 654B  E5           rup:    push hl                 ;save pointer to RUP row
3029 654C  0600                 ld b,0
3030 654E  3A0042               ld a,(me)
3031 6551  4F                   ld c,a                  ;make sure RUP(me) has i
3032 6552  09                   add hl,bc
3033 6553  7E                   ld a,(hl)
3034 6554  3C                   inc a
3035 6555  C403F0               call nz,rdie            ; ++ Another node same
3036 6558  E1                   pop hl                  ;restore RUP pointer
3037 6559  EB                   ex de,hl                ;move min DLY pointer to
3038 655A  FDE5                 push iy                 ;compute the base of LR
3039 655C  E1                   pop hl                  ;into HL
3040 655D  011D0C               ld bc,lrut
3041 6560  09                   add hl,bc               ;DE points to new, HL t
3042 6561  0600                 ld b,0                  ;start at node 0

3044 6563                       ;fall into RUPLOP
3045 6563
```

```
3046 6563                      ;from previous page

                       ;Get the old cost into C, the new one into A and update
                       ;in memory.

3051 6563  CDFD65      ruplop: call cost              ;compute the line's cost
3052 6566  4F                 ld c,a                 ;into c..
3053 6567  1A                 ld a,(de)              ;get new cost
3054 6568  FEFE               cp 0FEh                ;is it already infinite
3055 656A  3009               jr nc,ruplp2
3056 656C  81                 add a,c                ;no, increase it by new
3057 656D  3804               jr c,ruplpi            ;if overflow, make it in
3058 656F  FEFF               cp 0FFh                ;don't let it grow to FF
3059 6571  2002               jr nz,ruplp2           ;if it didn't, we're do
3060 6573  3EFE        ruplpi: ld a,0FEh             ;make it infinite
3061 6575  4E          ruplp2: ld c,(hl)              ;get old cost
3062 6576  77                 ld (hl),a              ;update old cost

3064 6577  B9                 cp c                   ;compare two costs
3065 6578  286E               jr z,rupnxt            ;if .EQ., no change in
3066 657A  301F               jr nc,rupgtr           ;if new cost .GT. old co

                       ;Here when new cost is less, check to see this new path
                       ;our MIN DLY, if it is, update it, too.

3071 657C  E5          ruplss: push hl                ;save HL
3072 657D  D5                  push de
3073 657E  21FB42              ld hl,costs            ;get ptr to cost table
3074 6581  1600                ld d,0                 ;make DE point to 0,B
3075 6583  58                  ld e,b                 ;(b = node number)
3076 6584  19                  add hl,de
3077 6585  BE                  cp (hl)                ;compare with MIN DLY
3078 6586  300F                jr nc,geqmin           ;if .GE. MIN, no need to
3079 6588  77                  ld (hl),a              ;update MIN dly
3080 6589  11F0FF              ld de,routes-costs     ;step to routing table
3081 658C  19                  add hl,de
3082 658D  56                  ld d,(hl)              ;get old line number
3083 658E  FD4E0D              ld c,(iy+lnum)         ;get line number
3084 6591  71                  ld (hl),c              ;update that, too
3085 6592  3EFF                ld a,0FFh              ;set NEWRUT
3086 6594  328742              ld (newrut),a

3088 6597  D1          geqmin: pop de                 ;all done
3089 6598  E1                 pop hl
3090 6599  184D               jr rupnxt              ;do another
3091 659B
```

```
                        ;Here when NEW delay is .GT. OLD delay. See if the old r
                        ;line that just got worse. If it was, re-compute the MI

3095 659B  E5           rupgtr: push hl                 ;save some reg
3096 659C  D5                   push de
3097 659D  21E842               ld hl,routes            ;get ptr to routing tabl
3098 65A0  1600                 ld d,0
3099 65A2  58                   ld e,b                  ;get node number
3100 65A3  19                   add hl,de
3101 65A4  FD7E0D               ld a,(iy+lnum)          ;this line?
3102 65A7  BE                   cp (hl)
3103 65A8  20ED                 jr nz,geqmin            ;if not, all done

                        ;The NEW delay is .GT. OLD delay and the path was out t
                        ;to look down the column of the NDT corresponding to thi
                        ;compute a new MIN and ROUTE. At this point, DE has the
                        ;Throughout this code, C has the "best so far" cost.

3110 65AA  C5           rupscn: push bc                 ;save a temp
3111 65AB  0EFF                 ld c,0FFh               ;assume un-reachable
3112 65AD  FDE5                 push iy                 ;save IY
3113 65AF  FD211B43             ld iy,lblk0             ;go through all the V-li

3115 65B3  FDCB0C66     scnlop: bit vbit,(iy+ltype1)    ;is this a V-link?
3116 65B7  2819                 jr z,scnnxt             ;if not, skip it
3117 65B9  FDE5                 push iy                 ;compute RUT address in
3118 65BB  E1                   pop hl
3119 65BC  D5                   push de                 ;save destination a mome
3120 65BD  111D0C               ld de,lrut
3121 65C0  19                   add hl,de
3122 65C1  D1                   pop de                  ;get destination off sta
3123 65C2  19                   add hl,de               ;compute RUT(destinati
3124 65C3  7E                   ld a,(hl)               ;get it
3125 65C4  B9                   cp c                    ;compare with best-so-fa
3126 65C5  2808                 jr z,scnnxt             ;if the same, skip it
3127 65C7  3009                 jr nc,scnnxt            ;or less than

3129 65C9  4F           scngtr: ld c,a                  ;update best
3130 65CA  21E842               ld hl,routes
3131 65CD  19                   add hl,de
3132 65CE  FD7E0D               ld a,(iy+lnum)          ;update it
3133 65D1  77                   ld (hl),a
3134 65D2
```

```
                        ;here to step to next row of NDT for this destination. T
                        ;next line block. When we run out, be sure to copy the
                        ;from C into COSTS(de).

3139 65D2  CDD664       scnnxt: call iynext             ;step to next
3140 65D5  38DC                 jr c,scnlop             ;loop until done
3141 65D7  21FB42               ld hl,costs
3142 65DA  19                   add hl,de               ;compute COSTS(line)
3143 65DB  71                   ld (hl),c               ;store best-so-far (coul
3144 65DC  3EFF                 ld a,0FFh               ;set flag for WATCH
3145 65DE  328742               ld (newrut),a
3146 65E1  1800                 jr scnpop

3148 65E3  FDE1         scnpop: pop iy                  ;restore IY
3149 65E5  C1                   pop bc                  ;and BC
3150 65E6  D1                   pop de                  ;then those pushed at RU
3151 65E7  E1                   pop hl

                        ;Here to do next routing update destination

3155 65E8  13           rupnxt: inc de                  ;step pointers
3156 65E9  23                   inc hl
3157 65EA  04                   inc b                   ;and count
3158 65EB  78                   ld a,b
3159 65EC  FE10                 cp nbox                 ;done?
3160 65EE  DA6365               jp c,ruplop             ;no - do next

3162 65F1  21FB42               ld hl,costs             ;get base of cost table.
3163 65F4  3A0042               ld a,(me)               ;load my box number.
3164 65F7  5F                   ld e,a
3165 65F8  1600                 ld d,0
3166 65FA  19                   add hl,de               ;index to my entry in t
3167 65FB  72                   ld (hl),d               ;zero cost getting to me
3168 65FC  C9                   ret
3169 65FD
```

```
                        ;Subroutine to compute the cost of a line into A.  Uses
                        ;Expects IY to have line number.

3173 65FD  DDE5         cost:   push ix                 ;save caller's IX
3174 65FF  E5                   push hl                 ;save some regs
3175 6600  D5                   push de
3176 6601  CDDE6A               call mpcrat             ;get the baud rate code.
3177 6604  5F                   ld e,a                  ;make DE
3178 6605  1600                 ld d,0                  ; have MRATE
3179 6607  211166               ld hl,ctab              ;get cost table
3180 660A  19                   add hl,de
3181 660B  7E                   ld a,(hl)
3182 660C  D1                   pop de                  ;restore some regs
3183 660D  E1                   pop hl
3184 660E  DDE1                 pop ix
3185 6610  C9                   ret

                        ;This is the routing cost table indexed by MRATE value (
                        ;F=19.2kb.  Each element is 19200/rate (thus for rate=48
                        ;Note that no value in this table should be .GE. 254 (F
                        ;are special values for the routine code.

3192 6611  FEFEAE8E     ctab:   defb 254,254,174,142
3193 6615  80402010             defb 128, 64, 32, 16
3194 6619  0A090805             defb  10,  9,  8,  5
3195 661D  04030201             defb   4,  3,  2,  1
3196 6621
```

```
                        ;This is the top of the Packet Processing Loop (PPL). Fi
                        ;the TO-BOX field which can have three values: for me,
                        ;routing packet which is destined for node FF.

                        ;First, clear the "blocked output" table, BLKTAB signi
                        ;output to any of the trunk lines. If at any time we can
                        ;to the trunk, we set BLKTAB(trunk)=FF to prevent shorte
                        ;out, thus violating the ordering of the messages.

3206 6621  21CB42       ppl:    ld hl,blktab            ;get a pointer to the bl
3207 6624  11CC42               ld de,blktab+1
3208 6627  011F00               ld bc,nlink-1           ;the table is NLINK long
3209 662A  3600                 ld (hl),0               ;zap the first entry by
3210 662C  EDB0                 ldir                    ;then do the rest

3212 662E  DD217542             ld ix,pplist            ;start at the top of the
3213 6632  DD4601       skip:   ld b,(ix+pforh)         ;get its forward point
3214 6635  DD4E00               ld c,(ix+pforl)
3215 6638  3E42         skip1:  ld a,pplist/256         ;check to see if it's ba
3216 663A  B8                   cp b
3217 663B  200E                 jr nz,pplmor            ;more to do if no match
3218 663D  3E75                 ld a,pplist&255
3219 663F  B9                   cp c
3220 6640  2009                 jr nz,pplmor            ;if match, we're done wi

3222 6642  3A8241       ppldon: ld a,(patrn)
3223 6645  E6CF                 and 0ffh-pkther-pktme
3224 6647  CD8169               call leds
3225 664A  C9                   ret

                        ;Here with valid new packet pointer in BC, move it to IX

3229 664B  C5           pplmor: push bc
3230 664C  DDE1                 pop ix
3231 664E  DD7E06               ld a,(ix+ptob)          ;get destination for th
3232 6651  FEFF                 cp 0FFh                 ;a routing packet?
3233 6653  CAA166               jp z,pplrup             ;yes - handle it
3234 6656  E67F                 and low7                ;remove sign
3235 6658  4F                   ld c,a                  ;copy destination to C
3236 6659  3A0042               ld a,(me)               ;compare with ME
3237 665C  B9                   cp c
3238 665D  CA0B67               jp z,pplme              ;if they're equal, it's

                        ;fall into PPLHIM
3241 6660
```

```
                        ;from previous page

3244 6660  3A8241   pplhim: ld a,(patrn)            ;load current light patt
3245 6663  F620             or pkther               ;store and forward packe
3246 6665  E6EF             and 0ffh-pktme
3247 6667  CD8169           call leds

3249 666A  0600             ld b,0                  ;make BC have destinati
3250 666C  21F842           ld hl,costs             ;index into COSTS to fin
3251 666F  09               add hl,bc
3252 6670  7E               ld a,(hl)               ;read reachability
3253 6671  FEFF             cp 0FFh                 ;is cost infinite?
3254 6673  CAA766           jp z,kill               ;yes, zap line

3256 6676  11F0FF           ld de,routes-costs      ;else step over to routi
3257 6679  19               add hl,de
3258 667A  7E               ld a,(hl)               ;fetch next hop

                        ;Enter at PPLOUT to send the packet pointed to by IX out
                        ;number in A. This is used primarily by RUPOUT.

3263 667B  CDDE64   pplout: call iylin              ;compute it's line point
3264 667E  0600             ld b,0
3265 6680  FD4E0D           ld c,(iy+lnum)          ;get line number
3266 6683  21C842           ld hl,blktab            ;see if that line is blo
3267 6686  09               add hl,bc               ;(B is still zero from
3268 6687  7E               ld a,(hl)
3269 6688  B7               or a
3270 6689  20A7             jr nz,skip              ;if BLKTAB <> 0, skip p

3272 668B  E5               push hl                 ;save BLKTAB address for

3274 668C  DDE5             push ix                 ;copy packet pointer to
3275 668E  E1               pop hl
3276 668F  010500           ld bc,plen              ;read PLEN word out of
3277 6692  09               add hl,bc
3278 6693  7E               ld a,(hl)
3279 6694  3C               inc a                   ;plus one to include it
3280 6695  0604             ld b,mwp                ;get a write packet comm
3281 6697  CD4A6A           call mpcout             ;output this packet to n

3283 669A  E1               pop hl                  ;get BLKTAB address off

3285 669B  380A             jr c,kill               ;if MPCOUT was a succes
3286 669D  36FF             ld (hl),0FFh            ;else, set BLKTAB entry
3287 669F  1891             jr skip                 ;and skip this packet


                        ;Here when there's a routing update (RUP) on the PPLIST.
                        ;an out-going RUP because PKTIN processes all incoming
                        ;has destination.

3294 66A1  DD7E04   pplrup: ld a,(ix+plink)         ;get destination
3295 66A4  C37B66           jp pplout               ;output it as a normal p
3296 66A7
```

```
                                ;Come to KILL to remove this packet from the PPLIST

3299 66A7  DD4601       kill:   ld b,(ix+pforh)         ;set up bc=next ptr
3300 66AA  DD4E00               ld c,(ix+porl)
3301 66AD  C5                   push bc
3302 66AE  CD0269               call givbuf             ;give it up
3303 66B1  C1                   pop bc
3304 66B2  C33866               jp skip1

                                ;Come to MOVE to move it to the end of the PPLIST

3308 66B5  C33266       move:   jp skip                 ;&& same as skip for now
3309 66B8
```

```
                        ;REL is run after each scheduling loop of VTN (much like
                        ;is responsible for sending RELEASE packets to any remo
                        ;them.  We withold RELEASEs when the local link is in CM
                        ;dataset conditions exist.  In the normal case, however,
                        ;for amount "k" is based on the following:
                        ;
                        ;       (255-LIN) + K  .LE.  PTL
                        ;
                        ;The term (255-LIN) represents the amount the sender can
                        ;window (he's sent LIN and therefore has 255-LIN left).
                        ;amount we're about to grant.  The sum of (255-LIN) and
                        ;or equal to the amount remaining in the MPC output buff
                        ;
                        ;       K .LE.  PTL-(255-LIN)
                        ;
                        ;To supress sending lots of very small "K"s, we only sen
                        ;K .GE. PTXNCH/2, half the size of one maximum length t
                        ;
                        ;NB: The usual two's complement issues apply to these co
                        ;is, one CANNOT reduce the second equation to PTL+LIN+1
                        ;is incorrect in that case.

3332 66B8  FD211843     rel:    ld iy,lblk0             ;start at line 0

3334 66BC  FDCB005E     rellop: bit onbit,(iy+ltype1)   ;line on?
3335 66C0  2843                 jr z,relnxt
3336 66C2  FD7E14               ld a,(iy+lflag1)        ;get its flags
3337 66C5  E6C0                 and cmdmsk+cnvmsk
3338 66C7  FE80                 cp 0+cnvmsk             ;does it make sense to
3339 66C9  203A                 jr nz,relnxt            ;if not, skip this
3340 66CB  FD7E1C               ld a,(iy+lin)           ;compute 255-LIN
3341 66CE  2F                   cpl                     ;(the fast way)
3342 66CF  47                   ld b,a                  ;save it
3343 66D0  CD036B               call mpcptl             ;compute PTL
3344 66D3  90                   sub b                   ;compute PTL-(255-LIN)
3345 66D4  382F                 jr c,relnxt             ;make sure PTL .GE. (255

3347 66D6  FE3D                 cp ptxnch/2             ;is it .GE. PTXNCH/2?
3348 66D8  382B                 jr c,relnxt             ;no, skip sending a RELE
3349 66DA  F5                   push af                 ;save amount on stack

3351 66DB  CDC468               call getbf0             ;try to get a buffer
3352 66DE  C1                   pop bc                  ;restore amount into B b
3353 66DF  D0                   ret nc                  ;if no buffers, stop nc

3355 66E0  DD360504             ld (ix+plen),prllen     ;set the length = RELEAS
3356 66E4  FD7E18               ld a,(iy+lremb)
3357 66E7  DD7706               ld (ix+ptob),a
3358 66EA  FD7E19               ld a,(iy+lreml)
3359 66ED  F680                 or 080H                 ;set sign indicating MT
3360 66EF  DD7707               ld (ix+ptol),a
3361 66F2  FD7E17               ld a,(iy+lcid)
3362 66F5  DD7708               ld (ix+pcid),a

                        ;The amount we release is in B

3366 66F8  DD7009               ld (ix+pminus),b
```

```
3367 66FB  FD7E1C            ld a,(iy+lin)          ;reduce LIN by that amou
3368 66FE  90                sub b
3369 66FF  DC03F0            call c,rdie            ; ++ REL: released morr
3370 6702  FD771C            ld (iy+lin),a

3372 6705  CDD664    relnxt: call iynext            ;step to next line bloc
3373 6708  38B2              jr c,rellop            ;loop
3374 670A  C9                ret                    ;all done
3375 670B
```

```
                        ;Here when the packet's for us. IX points to it. Check t
                        ;whether it's a SUBR packet or a TEXT/RELEASE

3379 670B  3A8241       pplme:  ld a,(patrn)            ;load current light patt
3380 670E  F610                 or pktme                ;the packet is for me,
3381 6710  E6DF                 and 0ffh-pkther         ; not store and forward.
3382 6712  CD8169               call leds

3384 6715  DDCB067E             bit 7,(ix+ptob)
3385 6719  C28567               jp nz,subr              ;if the sign is set, it'

                        ;If the sign is off, this packet is part of a conversati
                        ;to point to the place specified in the TOL field. Check
                        ;process the E bit.

3391 671C  DD7E07       pplcnv: ld a,(ix+ptol)          ;get the link number
3392 671F  E67F                 and low7                ;less the RELEASE pack
3393 6721  CDDE64               call iylin              ;map it into a line bloc
3394 6724  FDCB147E             bit cnvbit,(iy+lflag1)  ;in a connection?
3395 6728  CAA766               jp z,kill               ;nope, KILL
3396 672B  DD7E08               ld a,(ix+pcid)          ;get sequence
3397 672E  FDBE17               cp (iy+lcid)            ;a match?
3398 6731  C2A766               jp nz,kill              ;no, KILL

                        ;See if it's a RELEASE or TEXT

3402 6734  DDCB077E             bit 7,(ix+ptol)
3403 6738  CA5D67               jp z,text               ;if sign = 0, it's a tex

                        ;Here when it's a RELEASE from the remote.

3407 6738  DD7E05       cswind: ld a,(ix+plen)          ;make sure it's of the
3408 673E  FE04                 cp prllen
3409 6740  C403FC               call nz,rdie            ; ++ CSWIND: bad SWIND l
3410 6743  FD7E18               ld a,(iy+lout)          ;update LOUT
3411 6746  DD9609               sub (ix+pminus)
3412 6749  DC03FC               call c,rdie             ; ++ RELEASE more than L
3413 674C  FD7718               ld (iy+lout),a
3414 674F  FDCB1456             bit wwbit,(iy+lflag1)   ;is job waiting for a wi
3415 6753  CAA766               jp z,kill               ;no, done
3416 6756  FD361100            ld (iy+ltime),0         ;yes, wake him up now!
3417 675A  C3A766               jp kill                 ;and leave
3418 675D
```

```
                        ;Here when it's a TEXT packet.  Output the data and upda

3421 675D  DD7E05       text:   ld a,(ix+plen)          ;get the length
3422 6760  D603                 sub ptxovl              ;remove the overhead
3423 6762  CAA766               jp z,kill
3424 6765  DAA766               jp c,kill
3425 6768  47                   ld b,a                  ;save a copy of true len
3426 6769  FD861C               add a,(iy+lin)          ;update number of inst
3427 676C  DC03F0               call c,rdie             ; ++ TEXT: LIN overflow
3428 676F  FD771C               ld (iy+lin),a
3429 6772  78                   ld a,b                  ;restore length
3430 6773  DDE5                 push ix                 ;copy packet pointer to
3431 6775  E1                   pop hl
3432 6776  010900               ld bc,ptx1ch            ;get pointer to first c
3433 6779  09                   add hl,bc
3434 677A  0603                 ld b,mwt                ;get a write text functi
3435 677C  CD4A6A               call mpcout             ;output the text
3436 677F  D403F0               call nc,rdie            ; ++ TEXT: MPCOUT refusa
3437 6782  C3A766               jp kill                 ;kill the packet, proces
3438 6785
```

```
                        ;Here on SUBR packets.  Process the A-bit to see whether
                        ;ACK.  If it's a RQST, call the specified subroutine.
                        ;sure we're waiting for an ACK and if so, feed the packe
                        ;is waiting for it.

3444 6785  DD7E07       subr:   ld a,(ix+ptol)          ;get the TOL word
3445 6788  E63F                 and 03fH                ;remove the A-bit, leave
3446 678A  CDDE64               call iylin              ;set up a LINE
3447 678D  DDCB077E             bit 7,(ix+ptol)         ;is this an ACK or a RQS
3448 6791  2020                 jr nz,ack               ;if A=1, it's an ACK

                        ;Here when it's a request.  Load the registers and call

3452 6793  21A267               ld hl,sbrret            ;push a fake return add
3453 6796  E5                   push hl                 ;at the bottom of the st

3455 6797  DD660B               ld h,(ix+paddrh)        ;get hi addr
3456 679A  DD6E0A               ld l,(ix+paddrl)        ;get lo addr
3457 679D  E5                   push hl                 ;put it on the stack
3458 679E  CD6C68               call getreg             ;get the registers out
3459 67A1  C9                   ret                     ;"call" the subroutine.

                        ;It will return here...

3463 67A2  CD8768       sbrret: call setreg             ;set the registers into
3464 67A5  CDA268               call exchad             ;exchange the network a
3465 67A8  DDCB06FE             set 7,(ix+ptob)         ;set it to SUBR
3466 67AC  DDCB07FE             set 7,(ix+ptol)         ;set the A-bit
3467 67B0  C3B566               jp move                 ;and return it to the c
3468 67B3
```

```
                        ;Here when it's an ACK.  Check to be sure that the calle
                        ;ACK (ACKW set in LFLAG) and if all's well, reach up in
                        ;and fill in the remote's registers.

3473 67B3  DDCB0776     ack:    bit 6,(ix+ptol)         ;skip this packet?
3474 67B7  C23266               jp nz,skip              ;yes.
3475 67BA  FDCB1466             bit ackbit,(iy+1flag1)  ;waiting for ack?
3476 67BE  CC03FC               call z,rdie             ; ++ ACK: spurious ACK

3478 67C1  DDCB07F6             set 6,(ix+ptol)         ;set "leave alone" bit.
3479 67C5  ED736E42             ld (syssp),sp           ;save our stack pointer
3480 67C9  FD6610               ld h,(iy+1sp+1)         ;get RCALL's context
3481 67CC  FD6E0F               ld l,(iy+1sp+0)
3482 67CF  F9                   ld sp,hl
3483 67D0  E1                   pop hl                  ;get caller's IX off his
3484 67D1  FDCB146E             bit lrcbit,(iy+1flag1)  ;long RCALL?
3485 67D5  2803                 jr z,ack0               ;yes, IX will be packet
3486 67D7  DDE5                 push ix                 ;get packet address and
3487 67D9  E1                   pop hl                  ; it where it's expected
3488 67DA  227242       ack0:   ld (temp1),hl           ;save caller's (or pack
3489 67DD  F1                   pop af                  ;unwind HL,DE,BC,AF
3490 67DE  F1                   pop af
3491 67DF  F1                   pop af
3492 67E0  F1                   pop af
3493 67E1  CD6C68               call getreg             ;get context out of pack
3494 67E4  F5                   push af                 ;stash new context for
3495 67E5  C5                   push bc
3496 67E6  D5                   push de
3497 67E7  E5                   push hl
3498 67E8  FDCB14A6             res ackbit,(iy+1flag1)  ;wake up the ACKW job
3499 67EC  FD361100             ld (iy+1time),0         ;clear wait time
3500 67F0  2A7242               ld hl,(temp1)           ;get user's (or packet'
3501 67F3  E5                   push hl                 ;save IX in user's conte
3502 67F4  ED7B6E42             ld sp,(syssp)
3503 67F8  FDCB146E             bit lrcbit,(iy+1flag1)
3504 67FC  C23266               jp nz,skip              ;leave the packet around
3505 67FF  C3A766               jp kill                 ;kill packet.
3506 6802
```

```
                        ;Subroutine to do a remote call (RCALL).  Enter with PUS
                        ;of-subr-to-call.  Returns with these things popped off
                        ;if there's an error, else, PE (1).  If the long-RCALL b
                        ;assumed that the caller has a buffer already in IX.  Al
                        ;packet around with a pointer in IX if LRCBIT in LFLAG1

3513 6802  D9           rcall:  exx                     ;swap to alternate regis
3514 6803  08                   ex af,af'
3515 6804  E1                   pop hl                  ;find destination from s
3516 6805  D1                   pop de
3517 6806  C1                   pop bc
3518 6807  78                   ld a,b
3519 6808  CD0859               call nodeup             ;is it up?
3520 680B  304E                 jr nc,rerret            ;if not, return immedia
3521 680D  C5                   push bc
3522 680E  D5                   push de
3523 680F  E5                   push hl                 ;fix stack
3524 6810  FDCB146E             bit lrcbit,(iy+lflag1)  ;long RCALL?
3525 6814  2007                 jr nz,rcall0            ;yes, caller has set up
3526 6816                                               ; a standard length?)
3527 6816  CDDA68               call getbuf             ;no, get a buffer
3528 6819  DD36050F             ld (ix+plen),prclen     ;set length byte up
3529 681D  3A0042       rcall0: ld a,(me)               ;fill in the right valu
3530 6820  F680                 or 080h                 ;make it a SUBR
3531 6822  DD7708               ld (ix+pfrb),a
3532 6825  FD7E0D               ld a,(iy+lnum)          ;get our number
3533 6828  DD7709               ld (ix+pfrl),a          ;install it, too with si
3534 682B  08                   ex af,af'
3535 682C  D9                   exx                     ;get back caller's regi
3536 682D  CD8768               call setreg             ;put them into the buffe
3537 6830  E1                   pop hl                  ;get caller's PC off sta
3538 6831  C1                   pop bc                  ;get caller's ADDR off
3539 6832  DD7008               ld (ix+paddrh),b        ;put it into the packet
3540 6835  DD710A               ld (ix+paddrl),c
3541 6838  C1                   pop bc                  ;get destination off st
3542 6839  CBF8                 set 7,b                 ;force it to be a SUBR
3543 683B  CBB9                 res 7,c                 ;with the A-bit off
3544 683D  DD7006               ld (ix+ptob),b
3545 6840  DD7107               ld (ix+ptol),c
3546 6843  E5                   push hl                 ;push caller's address b
3547 6844
```

```
                        ;Here to sleep waiting for an ACK.

3550 6844  FDCB14E6             set ackbit,(iy+1flag1)  ;tell PPL we're waiting
3551 6848  061E                 ld b,30                 ;30 iterations gives 3 s
3552 684A  CD4C55       ackw:   call db01               ;wait 0.1 seconds
3553 684D  F5                   push af
3554 684E  FDCB1466             bit ackbit,(iy+1flag1)  ;get an ack?
3555 6852  2812                 jr z,acked              ;if the bit goes off, t
3556 6854  F1                   pop af
3557 6855  10F3                 djnz ackw               ;else loop

3559 6857  AF           poret:  xor a                   ;get a zero, clearing P
3560 6858  F601                 or 1                    ;set P
3561 685A  C9                   ret                     ;return with PO

3563 685B  E5           rerret: push hl                 ;put caller's PC back
3564 685C  D9                   exx                     ;get his regs back
3565 685D  FDCB146E             bit lrcbit,(iy+1flag1)
3566 6861  C40269               call nz,givbuf
3567 6864  18F1                 jr poret                ;(no need to restore AF

                        ;Here when it was ACK'ed.  PPL has fudged our registers
                        ;set the PE bit for success (it's bit 2 of the F-reg) a

3572 6866  E3           acked:  ex (sp),hl              ;exchange AF (from stack
3573 6867  CBD5                 set 2,l                 ;set the Parity Even (P'
3574 6869  E3                   ex (sp),hl              ;restore modified AF to
3575 686A  F1                   pop af                  ;pop it.
3576 686B  C9                   ret                     ;return to caller
3577 686C
```

```
                        ;Subroutine to read the registers out of a SUBR packet

3580 686C DD6612        getreg: ld h,(ix+pregh)         ;restore the registers
3581 686F DD6E13                ld l,(ix+pregl)
3582 6872 DD5E11                ld e,(ix+prege)
3583 6875 DD5610                ld d,(ix+pregd)
3584 6878 DD460C                ld b,(ix+prega)         ;read AF into BC for a m
3585 687B DD4E0D                ld c,(ix+pregf)
3586 687E C5                    push bc                 ;then copy BC into AF
3587 687F F1                    pop af
3588 6880 DD460E                ld b,(ix+pregb)         ;then restore the real
3589 6883 DD4E0F                ld c,(ix+pregc)
3590 6886 C9                    ret                     ;and return into the sub

                        ;Subroutine to store the registers in the SUBR packet

3594 6887 DD7412        setreg: ld (ix+pregh),h
3595 688A DD7513                ld (ix+pregl),l
3596 688D DD7210                ld (ix+pregd),d
3597 6890 DD7311                ld (ix+prege),e
3598 6893 DD700E                ld (ix+pregb),b
3599 6896 DD710F                ld (ix+pregc),c
3600 6899 F5                    push af                 ;copy AF into BC
3601 689A C1                    pop bc
3602 689B DD700C                ld (ix+prega),b
3603 689E DD710D                ld (ix+pregf),c
3604 68A1 C9                    ret

                        ;Subroutine to flip the TOB/TOL with the FRB/FRL words
                        ;sign bits to zero.

3609 68A2 DD5606        exchad: ld d,(ix+ptob)
3610 68A5 DD5E07                ld e,(ix+ptol)
3611 68A8 DD6608                ld h,(ix+pfrb)
3612 68AB DD6E09                ld l,(ix+pfrl)
3613 68AE EB                    ex de,hl
3614 68AF CBBA                  res 7,d
3615 68B1 CBBB                  res 7,e
3616 68B3 CBBC                  res 7,h
3617 68B5 CBBD                  res 7,l
3618 68B7 DD7206                ld (ix+ptob),d
3619 68BA DD7307                ld (ix+ptol),e
3620 68BD DD7408                ld (ix+pfrb),h
3621 68C0 DD7509                ld (ix+pfrl),l
3622 68C3 C9                    ret
3623 68C4
```

```
                        ;Subroutines to manipulate buffers.
                        ;GETBUF move from FREE to PPLIST, blocking as necessary
                        ;GETBF0 move from FREE to PPLIST, non-blocking (sets C/N
                        ;GIVBUF move from PPLIST to FREE (never blocks)

3629 68C4  2A7D42       getbf0: ld hl,(frleng)          ;look at the free list l
3630 68C7  7C                   ld a,h
3631 68C8  B5                   or l
3632 68C9  C8                   ret z                   ;if empty, punt with C=0
3633 68CA  CDE868               call getbff             ;else get one
3634 68CD  37                   scf
3635 68CE  C9                   ret

3637 68CF  3A8241       getbf1: ld a,(patrn)            ;get current light pat
3638 68D2  F604                 or buffw
3639 68D4  CD8169               call leds
3640 68D7  CD3E55               call db                 ;else wait
3641 68DA  CDC468       getbuf: call getbf0             ;try to get one
3642 68DD  30F0                 jr nc,getbf1            ;cant - loop
3643 68DF  3A8241               ld a,(patrn)
3644 68E2  E6FB                 and 0ffh-buffw          ;not waiting for a buffe
3645 68E4  CD8169               call leds
3646 68E7  C9                   ret

3648 68E8  2A7D42       getbff: ld hl,(frleng)          ;prepare to decr count
3649 68EB  2B                   dec hl
3650 68EC  227D42               ld (frleng),hl
3651 68EF  CB7C                 bit 7,h                 ;did hi order go negativ
3652 68F1  C403FC               call nz,rdie            ; ++ FRLENG went negati
3653 68F4  DD2A7942             ld ix,(frlist)          ;cell to be removed is f
3654 68F8  CD1369               call deq
3655 68FB  217542               ld hl,pplist            ;ptr to destination
3656 68FE  CD5C69               call enq                ;enq it there
3657 6901  C9                   ret

3659 6902  CD1369       givbuf: call deq
3660 6905  217942               ld hl,frlist            ;get ptr to free list
3661 6908  CD5C69               call enq
3662 690B  2A7D42               ld hl,(frleng)
3663 690E  23                   inc hl
3664 690F  227D42               ld (frleng),hl
3665 6912  C9                   ret                     ;return
3666 6913
```

```
                        ;Subroutine to DEQ some cell from its queue. Enter with
                        ;pointing to cell.

3670 6913 DD6603        deq:    ld h,(ix+pbakh)         ;get ptr to previous in
3671 6916 DD6E02                ld l,(ix+pbakl)

3673 6919                       if debbuf               ;if debugging buffers
3674 6919 FDE5                  push iy                 ;** save IY (sometimes
3675 691B E5                    push hl                 ;** move HL to
3676 691C FDE1                  pop iy                  ;** an index register.
3677 691E DDE5                  push ix                 ;** move pointer to cu
3678 6920 D1                    pop de                  ;** to friendlier ac pai
3679 6921 FD7E01                ld a,(iy+pforh)         ;** check to see whether
3680 6924 BA                    cp d                    ;** of last buffer is t
3681 6925 C403FC                call nz,rdie            ;** back pointer of this
3682 6928 FD7E00                ld a,(iy+pforl)         ;** (now low halves).
3683 692B BB                    cp e                    ;** ...
3684 692C C403FC                call nz,rdie            ;** ...
3685 692F FDE1                  pop iy                  ;** restore IY.
3686 6931                       endif

3688 6931 DD5601                ld d,(ix+pforh)         ;get ptr to next into DE
3689 6934 DD5E00                ld e,(ix+pforl)

3691 6937                       if debbuf
3692 6937 E5                    push hl                 ;** save pointer to pre
3693 6938 FDE5                  push iy                 ;** get an indexable wor
3694 693A D5                    push de                 ;** move ac pair to
3695 693B FDE1                  pop iy                  ;** the index ac.
3696 693D DDE5                  push ix                 ;** copy current pointer
3697 693F E1                    pop hl                  ;** better ac pair.
3698 6940 FD7E03                ld a,(iy+pbakh)         ;** point back
3699 6943 BC                    cp h                    ;** to see if
3700 6944 C403FC                call nz,rdie            ;** pointers
3701 6947 FD7E02                ld a,(iy+pbakl)         ;** match...
3702 694A BD                    cp l                    ;** ...
3703 694B C403FC                call nz,rdie            ;** ...
3704 694E FDE1                  pop iy                  ;** all is ok, restore
3705 6950 E1                    pop hl                  ;** ...
3706 6951                       endif

3708 6951 73                    ld (hl),e               ;set up PFORL of previou
3709 6952 23                    inc hl
3710 6953 72                    ld (hl),d
3711 6954 2B                    dec hl
3712 6955 EB                    ex de,hl

3714 6956 23                    inc hl                  ;point to PFORH
3715 6957 23                    inc hl                  ;point to PBAKL
3716 6958 73                    ld (hl),e               ;set up PBAKL of next
3717 6959 23                    inc hl
3718 695A 72                    ld (hl),d

3720 695B C9                    ret
3721 695C
```

```
                        ;Subroutine to ENQ a new cell into a queue. Enter with I
                        ;one which will become its predessor (i.e. we ENQ BEFOR
3725 695C DD7401        enq:    ld (ix+pforh),h         ;set us to point to new
3726 695F DD7500                ld (ix+porl),l

3728 6962 23                    inc hl                  ;point to PFORH of new n
3729 6963 23                    inc hl                  ;point to PBAKL of new
3730 6964 5E                    ld e,(hl)               ;get ptr to new previous
3731 6965 23                    inc hl
3732 6966 56                    ld d,(hl)

3734 6967 DD7203                ld (ix+pbakh),d         ;point our back to his o
3735 696A DD7302                ld (ix+pbakl),e

3737 696D DDE5                  push ix                 ;get our address into
3738 696F C1                    pop bc                  ;a useful place

3740 6970 70                    ld (hl),b               ;hl points to new next's
3741 6971 2B                    dec hl
3742 6972 71                    ld (hl),c               ;now to its PBAKL

3744 6973 62                    ld h,d                  ;get new previous's add
3745 6974 6B                    ld l,e

3747 6975 71                    ld (hl),c               ;points to new previous
3748 6976 23                    inc hl
3749 6977 70                    ld (hl),b               ;PFORH

3751 6978 C9                    ret
3752 6979
```

```
                        ;Subroutine to return a new sequence number for connecti

3755 6979  211B53       gcid:   ld hl,cid
3756 697C  34           gcid1:  inc (hl)
3757 697D  7E                   ld a,(hl)
3758 697E  28FC                 jr z,gcid1
3759 6980  C9                   ret

                        ;Subroutine to output A to the LIGHTS and save value in
                        ;de-selects any MPC.

3765 6981  D3FE         leds:   out (lights),a
3766 6983  328241               ld (patrn),a
3767 6986  3EFF                 ld a,mpc99
3768 6988  D3FF                 out (select),a
3769 698A  C9                   ret
3770 698B
```

```
                        ;MPC interface routines (fairly low-level).

3773 698B  FD7E0D       iynum:  ld a,(iy+lnum)          ;pick out the line numbe

                        ;MPCNUM computes the MPC's hardware address from a line
                        ;result in A, no ACs changed. IYNUM also selects line nu

3778 698E  1F           mpcnum: rra                     ;div 2
3779 698F  1F                   rra                     ;div 4, possibly leaving
3780 6990  C6F0                 add a,mpc0              ;offset low 4 bits to re
3781 6992  F6F0                 or 0f0h                 ;force it to be in rang
3782 6994  C9                   ret

                        ;MPCSEL select the MPC belonging to line number in this
                        ;Uses AC A only.

3787 6995  CD8B69       mpcsel: call iynum              ;compute port number in
3788 6998  D3FF         mpcsla: out (select),a          ;select this port
3789 699A  3A660C               ld a,(066h)             ;make sure it got select
3790 699D  FEC3                 cp 0C3h                 ; ... and making sure it
3791 699F  C8                   ret z                   ;finally made it.
3792 69A0  CD03FC               call rdie               ; ++ MPCSEL: MPC failed

                        ;MPCUNS de-selects the current MPC by selecting "MPC99",
                        ;Note that MPCUNS does not destory AF

3797 69A3  F5           mpcuns: push af

3799 69A4  3A8241               ld a,(patrn)
3800 69A7  E6F7                 and 0ffh-iopw           ;can't be waiting for I/
3801 69A9  CD8169               call leds

3803 69AC  F1                   pop af
3804 69AD  C9                   ret                     ;done

                        ;UNSCHK will check to be sure no MPC is selected. If one
                        ;Uses AF only.

3809 69AE  3A000C       unschk: ld a,(0)                ;make sure zero is FF (?
3810 69B1  3C                   inc a
3811 69B2  C8                   ret z
3812 69B3  C403FC               call nz,rdie            ; ++ UNSCHK: MPC failed
3813 69B6
```

```
                        ;MPCWAT waits for this MPC (number via line block) to fi
                        ;This is a little tricky; we use two loops.  The outer
                        ;to see if the MPC is finished.  The inner (faster) loop
                        ;to see if it has changed lately.  Specifically, the sig
                        ;toggle upon command completion.  The faster loop norma
                        ;completion okay, but to be safe, we have the outer guar
                        ;don't race with the MPC.  Returns with MPC selected!

3822 69B6  C5           mpcwat: push bc                 ;save a temp
3823 69B7  D5                   push de                 ;save another temp
3824 69B8  CD8B69               call iynum              ;compute MPC number fr
3825 69BB  4F                   ld c,a                  ;move it to useful place
3826 69BC  ED40                 in b,(c)                ;remember original SPORT

3828 69BE  3A8241               ld a,(patrn)            ;load light pattern.
3829 69C1  F608                 or iopw
3830 69C3  CD8169               call leds
3831 69C6  1810                 jr mpcwte               ;start at outer loop fir

3833 69C8  3EFF         mpcwtl: ld a,mpc99              ;get impossible MPC
3834 69CA  D3FF                 out (select),a          ;select it, de-selecting
3835 69CC                       ;This will start MPC run

                        ;inner loop... 32 tries for SPORT change

3839 69CC  1E20                 ld e,32                 ;start here
3840 69CE  ED78         mpcwti: in a,(c)                ;read new SPORT
3841 69D0  A8                   xor b                   ;did command status chan
3842 69D1  E67F                 and low7                ; (it is the top bit)
3843 69D3  2003                 jr nz,mpcwte            ;SPORT changed, make sur
3844 69D5  1D                   dec e                   ;more to iterate?
3845 69D6  20F6                 jr nz,mpcwti            ;if yes, go iterate
3846 69D8  CD9569       mpcwte: call mpcsel             ;select it
3847 69DB  3A002C               ld a,(mvc)              ;look at command
3848 69DE  3D                   dec a                   ;less one
3849 69DF  FAE469               jp m,mpcwtr             ;if was negative or went

3851 69E2  18E4                 jr mpcwtl               ;and try outer loop aga

3853 69E4  D1           mpcwtr: pop de                  ;fix stack
3854 69E5  C1                   pop bc                  ;restore BC
3855 69E6  3C                   inc a                   ;fix A and set more CCR
3856 69E7  C9                   ret
3857 69E8
```

```
                        ;MPCDO will do a command for you.  Enter with A=MV1 and
                        ;pointing to reasonable line block from which to fetch
                        ;Returns with MPC selected (from MPCWAT).

3862 69E8  4F           mpcdo:  ld c,a                  ;save A
3863 69E9  CD9569               call mpcsel             ;select this MPC
3864 69EC  3A002C               ld a,(mvc)              ;make sure previous comm
3865 69EF  3D                   dec a
3866 69F0  F403FC               call p,rdie             ; ++ MPCDO overran MPC

3868 69F3  E5                   push hl                 ;save a temp
3869 69F4  21002C               ld hl,mvc               ;get to first word
3870 69F7  70                   ld (hl),b               ;store MVC
3871 69F8  23                   inc hl                  ;step to MVL
3872 69F9  FD7E0D               ld a,(iy+lnum)
3873 69FC  E603                 and 3
3874 69FE  77                   ld (hl),a               ;store MVL
3875 69FF  23                   inc hl                  ;step to MV1
3876 6A00  71                   ld (hl),c               ;store MV1
3877 6A01  E1                   pop hl

                        ;Caller may enter here if:
                        ;       1)      MPC already selected.
                        ;       2)      new MVC stored.
                        ;       3)      new MV1 stored.

3884 6A02  CDB669       mpcdos: call mpcwat             ;wait for it to finish t
3885 6A05  CB77                 bit mftbit,a            ;is it fatal?
3886 6A07  C403FC               call nz,rdie            ; ++ Fatal MPC error
3887 6A0A  B7                   or a                    ;fix CCR again
3888 6A0B  C9                   ret
3889 6A0C
```

```
                        ;MPCIN reads text into VTN memory. Enter with A=max desi
                        ;to recipient buffer area and B=command to issue to MPC
                        ;Returns with actual length in A and CCR, BC,HL destroye

3894 6A0C  B7           mpcin:  or a                    ;make sure max length
3895 6A0D  CC03FC               call z,rdie             ; ++ MPCIN: bad length a
3896 6A10  CDE869               call mpcdo              ;do this command
3897 6A13  280A                 jr z,mpcin2             ;if zero, fine
3898 6A15  D684                 sub mmtyer              ;if empty, set A=0
3899 6A17  CAA369               jp z,mpcuns             ;release MPC and return.
3900 6A1A  C684                 add a,mmtyer            ;fix A for crash
3901 6A1C  CD03FC               call rdie               ; ++ MPCIN saw bad MPC r

3903 6A1F  D5           mpcin2: push de                 ;save DE for LDIRs

3905 6A20  EB                   ex de,hl                ;put buffer address into
3906 6A21  2A052C               ld hl,(mvx1a)           ;get first part xfer ad
3907 6A24  ED4B0720             ld bc,(mvx1c)           ;and count
3908 6A28  EDB0                 ldir                    ;copy it

3910 6A2A  2A092C               ld hl,(mvx2a)           ;get second part
3911 6A2D  ED4B0820             ld bc,(mvx2c)           ;and count
3912 6A31  78                   ld a,b
3913 6A32  B1                   or c                    ;any second part?
3914 6A33  2802                 jr z,mpcin3             ;if not, skip second ldi
3915 6A35  EDB0                 ldir                    ;else do it

3917 6A37  D1           mpcin3: pop de                  ;restore DE
3918 6A38  3A022C               ld a,(mv1)              ;get actual length
3919 6A3B  B7                   or a                    ;set CCR
3920 6A3C  F5                   push af                 ;save it for our caller
3921 6A3D  0605                 ld b,mer                ;for MPCDO. set up b=er
3922 6A3F  CDE869               call mpcdo              ;do this command
3923 6A42  C403FC               call nz,rdie            ; ++ MPCIN saw illegal r
3924 6A45  CDA369               call mpcuns             ;de-select it
3925 6A48  F1                   pop af                  ;get back MV1 and flags
3926 6A49  C9                   ret
3927 6A4A
```

```
                        ;MPCOUT has similar args as MPCIN except that returns ca
                        ;carry if output succeeds or fails (no partial output )
                        ;with A=length, HL to send buffer area and B=command to
                        ;(MWT or MWP). BC,HL destroyed.

3933 6A4A  B7           mpcout: or a                    ;make sure caller knows
3934 6A4B  CC03FC               call z,rdie             ; ++ MPCOUT: bad length
3935 6A4E  CDE869               call mpcdo              ;do this command
3936 6A51  280A                 jr z,mpcou2             ;if zero, fine
3937 6A53  EE82                 xor mnrmer              ;if no room, set A=0 and
3938 6A55  CAA369               jp z,mpcuns             ;return with flag, rel
3939 6A58  EE82                 xor mnrmer              ;fix A for crash
3940 6A5A  CD03FC               call rdie               ; ++ MPCOUT saw bad MPC

3942 6A5D  D5           mpcou2: push de                 ;save DE for LDIRs
3943 6A5E  ED5B0520             ld de,(mvx1a)           ;get first part xfer add
3944 6A62  ED4B0720             ld bc,(mvx1c)           ;and count
3945 6A66  EDB0                 ldir                    ;copy it

3947 6A68  ED5B0920             ld de,(mvx2a)           ;get second part
3948 6A6C  ED4B0B20             ld bc,(mvx2c)           ;and count
3949 6A70  78                   ld a,b
3950 6A71  B1                   or c                    ;any second part?
3951 6A72  2802                 jr z,mpcou3             ;if not, skip second ldi
3952 6A74  EDB0                 ldir                    ;else do it

3954 6A76  D1           mpcou3: pop de                  ;restore DE
3955 6A77  0606                 ld b,mew                ;for MPCDO. set up b=end
3956 6A79  3A022C               ld a,(mv1)              ;mimic size arg back to
3957 6A7C  CDE869               call mpcdo              ;do this command
3958 6A7F  C403FC               call nz,rdie            ; ++ MPCOUT saw illegal
3959 6A82  CDA369               call mpcuns             ;de-select it
3960 6A85  37                   scf                     ;set carry for good retu
3961 6A86  C9                   ret
3962 6A87
```

```
                        ;MPCIC gets a character from the MPC into A.  Does not a
                        ;Makes room for the read by using space on the stack.
                        ;returns with carry=0.

3967 6A87  CD3068       mpcic:  call active             ;any activity on this l
3968 6A8A  C8                   ret z                   ;no, return (carry off).

3970 6A8B  E5                   push hl                 ;save a temp
3971 6A8C  E5                   push hl                 ;now open the stack by o
3972 6A8D  3E01                 ld a,1                  ;size=1
3973 6A8F  210000               ld hl,0                 ;load zero into HL.
3974 6A92  39                   add hl,sp               ;copy stack pointer.
3975 6A93  C5                   push bc                 ;make room for command i
3976 6A94  0601                 ld b,mrt                ;read text
3977 6A96  CD0C6A               call mpcin              ;attempt to read it
3978 6A99  C1                   pop bc                  ;restore stack (BC)
3979 6A9A  2806                 jr z,mpcice             ;failed
3980 6A9C  E1                   pop hl                  ;read the buffer word fr
3981 6A9D  7D                   ld a,l                  ;get character out of it
3982 6A9E  B7                   or a                    ;set Z/NZ for caller
3983 6A9F  E1                   pop hl                  ;restore HL
3984 6AA0  37                   scf                     ;give good return
3985 6AA1  C9                   ret                     ;and return

3987 6AA2  E1           mpcice: pop hl                  ;fix stack
3988 6AA3  E1                   pop hl
3989 6AA4  AF                   xor a                   ;clear carry
3990 6AA5  C9                   ret                     ;bad return
3991 6AA6
```

```
                        ;MPCOC plays a similar game.

3994 6AA6  E5           mpcoc:  push hl                 ;save HL
3995 6AA7  6F                   ld l,a                  ;A on stack
3996 6AA8  E5                   push hl
3997 6AA9  210000               ld hl,0
3998 6AAC  39                   add hl,sp               ;load stack pointer in H
3999 6AAD  3E01                 ld a,1                  ;length=1
4000 6AAF  C5                   push bc                 ;make room for command
4001 6AB0  0603                 ld b,mwt                ;write text
4002 6AB2  CD4A6A               call mpcout             ;try to output it setti
4003 6AB5  C1                   pop bc
4004 6AB6  E1                   pop hl
4005 6AB7  E1                   pop hl
4006 6AB8  C9                   ret

                        ;MPCBKA sets the NZ bit if this user has typed a break
                        ;On return B0 of A=break bit, B1=attn bit

4011 6AB9  CD306B       mpcbka: call active             ;see if any activity or
4012 6ABC  C8                   ret z                   ;if status is nil, retur
4013 6ABD  C5                   push bc                 ;musn't trash any ac's
4014 6ABE  0619                 ld b,mrcbk              ;read and clear break
4015 6AC0  CDE869               call mpcdo
4016 6AC3  C403FC               call nz,rdie            ; ++ MPCBKA: can't read
4017 6AC6  3A0220               ld a,(mv1)              ;load bits.
4018 6AC9  B7                   or a
4019 6ACA  C1                   pop bc
4020 6ACB  C3A369               jp mpcuns               ;deselect MPC and retur

                        ;MPCSTA reads status bits of a line.

4024 6ACE  C5           mpcsta: push bc                 ;don't trash the ACs.
4025 6ACF  060C                 ld b,mr3
4026 6AD1  CDE869               call mpcdo
4027 6AD4  C403FC               call nz,rdie
4028 6AD7  3A0420               ld a,(mv3)              ;load bits
4029 6ADA  C1                   pop bc
4030 6ADB  C3A369               jp mpcuns
4031 6ADE
```

```
                        ;MPCRAT returns the baud rate of the line in A, FF if hu
4034 6ADE C5            mpcrat: push bc
4035 6ADF 060D                  ld b,mrbaud             ;read the
4036 6AE1 CDE869                call mpcdo              ; baud rate
4037 6AE4 C403FC                call nz,rdie            ; ++ MPCRAT: can't read
4038 6AE7 3A0220                ld a,(mv1)              ;load baud rate code.
4039 6AEA C1                    pop bc
4040 6AEB C3A369                jp mpcuns               ;deselect MPC and return

                        ;MPCKBC returns the KB character counter in A and CCR.

4044 6AEE CD306B        mpckbc: call active             ;if SPORT indicates noth
4045 6AF1 C8                    ret z                   ; return...
4046 6AF2 C5                    push bc
4047 6AF3 060C                  ld b,mr3                ;read KCNT, PCNT, status
4048 6AF5 CDE869                call mpcdo
4049 6AF8 C403FC                call nz,rdie            ; ++ MPCKBC: can't read
4050 6AFB 3A0220                ld a,(mv1)              ;load keyboard count.
4051 6AFE B7                    or a                    ;set condition code.
4052 6AFF C1                    pop bc
4053 6B00 C3A369                jp mpcuns               ;deselect MPC and return

                        ;MPCPTL returns the number of characters left in the pri

4057 6B03 C5            mpcptl: push bc
4058 6B04 060C                  ld b,mr3                ;read KCNT, PCNT
4059 6B06 CDE869                call mpcdo              ;...
4060 6B09 C403FC                call nz,rdie            ; ++ MPCPTL: can't read
4061 6B0C 3A0320                ld a,(mv2)              ;load PCNT
4062 6B0F EEFF                  xor 0ffh                ;compute 255-x, (CPL doe
4063 6B11 C1                    pop bc                  ;don't destroy BC.
4064 6B12 C3A369                jp mpcuns               ;deselect MPC and return
4065 6B15
```

```
                        ;MPCNMI will NMI the MPC pointed to by IY (LNUM). Note &
                        ;there's something of a race here (around MPCWAT). It
                        ;while after NMI'ing the MPC.

4070 6B15  C5           mpcnmi: push bc                 ;save a temp
4071 6B16  CD9569               call mpcsel             ;get the mpc (even if it
4072 6B19  3E08                 ld a,mrn                ;get a restart no-op (pr
4073 6B1B  32002C               ld (mvc),a              ;force MVC positive fo
4074 6B1E  CD8B69               call iynum              ;get the port number int
4075 6B21  4F                   ld c,a                  ;move to useful place
4076 6B22  ED79                 out (c),a               ;NMI this MPC

4078 6B24  CDB669               call mpcwat             ;wait for it to finish
4079 6B27  FE85                 cp mrster               ;better have restarted
4080 6B29  C403FC               call nz,rdie            ; ++ MPCRLD: can't NMI M

4082 6B2C  C1                   pop bc                  ;restore stack
4083 6B2D  C3A369               jp mpcuns

                        ;ACTIVE tests for SPORT activity for a given line. A i

4087 6B30  C5           active: push bc
4088 6B31  CD8B69               call iynum              ;get i/o address (Fx)
4089 6B34  4F                   ld c,a                  ;copy to i/o ac.
4090 6B35  ED78                 in a,(c)                ;read status port.
4091 6B37  C1                   pop bc
4092 6B38  FDA60E               and (iy+lmymsk)         ;see if my line is activ
4093 6B3B  C9                   ret
4094 6B3C
```

```
                        ;MPCZPL will ask the MPC to reset the line.

4097 6B3C  F5           mpczpl: push af
4098 6B3D  C5                   push bc

4100 6B3E  AF                   xor a                   ;start with offline code
4101 6B3F  FDCB005E             bit onbit,(iy+ltype1)   ;is the line in operatio
4102 6B43  2818                 jr z,mpcnv
4103 6B45  3C                   inc a                   ;prepare to send H-link/
4104 6B46  FDCB0066             bit vbit,(iy+ltype1)    ;is it a V-link?
4105 6B4A  2801                 jr z,mpcsh
4106 6B4C  3C                   inc a                   ;load V-Link code (2)

4108 6B4D  FDCB004E     mpcsh:  bit shbit,(iy+ltype1)   ;shaking?
4109 6B51  2802                 jr z,mpcnsh
4110 6B53  F680                 or 80h                  ;light bit

4112 6B55  FDCB0046     mpcnsh: bit dcebit,(iy+ltype1)  ;DCE/DTE?
4113 6B59  2802                 jr z,mpcnv
4114 6B5B  F640                 or 40h                  ;light DCE bit

4116 6B5D  0607         mpcnv:  ld b,mkl                ;kill line, load "new" c
4117 6B5F  CDE869               call mpcdo
4118 6B62  C403FC               call nz,rdie            ; ++ MPCZPL: Can't reset

4120 6B65  FD7E03               ld a,(iy+lclip)         ;get the keyboard clip
4121 6B68  0618                 ld b,mwclip             ;load "write clip" comma
4122 6B6A  CDE869               call mpcdo              ;do this command
4123 6B6D  C403FC               call nz,rdie            ; ++ MPCZPL: can't set

4125 6B70  FD7E0A               ld a,(iy+lrate)         ;baud rate.
4126 6B73  060E                 ld b,mwbaud
4127 6B75  CDE869               call mpcdo
4128 6B78  C403FC               call nz,rdie            ; ++ MPCZPL: can't set b

4130 6B7B  FD7E08               ld a,(iy+lpad)          ;padding
4131 6B7E  061C                 ld b,mwpad
4132 6B80  CDE869               call mpcdo
4133 6B83  C403FC               call nz,rdie

4135 6B86  FD7E0C               ld a,(iy+lintr)         ;interrupt character
4136 6B89  061E                 ld b,mwint
4137 6B8B  CDE869               call mpcdo
4138 6B8E  C403FC               call nz,rdie
4139 6B91
```

```
4141 6B91  AF                    xor a
4142 6B92  0614                  ld b,mwpx                ;set process flow
4143 6B94  FDCB0176              bit pfxbit,(iy+ltype2)   ; if bit is set.
4144 6B98  CDC66B                call setnz

4146 6B9B  AF                    xor a
4147 6B9C  0616                  ld b,mwgx                ;set generate flow
4148 6B9E  FDCB017E              bit gfxbit,(iy+ltype2)   ; if bit is set.
4149 6BA2  CDC66B                call setnz

4151 6BA5  FDCB0C66              bit vbit,(iy+ltype1)     ;this a V-link?
4152 6BA9  2015                  jr nz,novl               ;not legal to set parity

4154 6BAB  FD7E09                ld a,(iy+lprty)          ;set parity.
4155 6BAE  0612                  ld b,mwprty
4156 6BB0  CDE869                call mpcdo
4157 6BB3  C403FC                call nz,rdie

4159 6BB6  AF                    xor a                    ;prepare to send a zero
4160 6BB7  0610                  ld b,mwecho              ; for echo status (full-
4161 6BB9  FDCB157E              bit eclbit,(iy+lflag2)   ;should there be echo?
4162 6BBD  CDC66B                call setnz               ;will set to zero if ec
4163 6BC0                                                 ;set to one.
4164 6BC0  CDA369          novl: call mpcuns              ;finally, deselect board
4165 6BC3  C1                    pop bc
4166 6BC4  F1                    pop af
4167 6BC5  C9                    ret

                           ;this subroutine sends A (typically zero) or A+1 to the
                           ;done by the caller.  B must contain MPC command dispatc

4172 6BC6  2801            setnz: jr z,setnz0
4173 6BC8  3C                    inc a
4174 6BC9  CDE869          setnz0: call mpcdo
4175 6BCC  C8                    ret z
4176 6BCD  C303FC                jp rdie
4177 6BD0
```

```
0 6BD0                  *include d.z80
                        ;This file contains the additional code to make a VBOX
                        ;commands and disk support.

4 000D                  dsize:  equ ldend-ldbeg+1       ;size of DBLK image
5 6BD0                  dblk:   defs dsize
6 6BDD
```

```
                        ;XCLHI is the "high-memory" portion of XCL.  That is, XC
                        ;XCLHI if the machine has this module loaded.  If not, 
                        ;error to the user.

   11 6BDD  CD6B63      xclhi:  call dscchk             ;must be disconnected
   12 6BE0  CDC463              call getlok             ;get the XCL command int
   13 6BE3  CDA259              call dispth
   14 6BE6  41                  defb 'A'
   15 6BE7  D36CF86B            defw altr,alttxt
   16 6BEB  4A                  defb 'J'
   17 6BEC  626DDC6C            defw jump,jmptxt
   18 6BF0  53                  defb 'S'
   19 6BF1  CD6E3C6C            defw pst,psttxt
   20 6BF5  57                  defb 'W'
   21 6BF6  B86D1C6C            defw watch,wattxt
   22 6BFA  FF                  defb -1

   24 6BFB  6C746572    alttxt: defm 'lter',0
   24 6BFF  00
   25 6C00  756D7020    jmptxt: defm 'ump to bootstrap at (node) ',0
   25 6C04  746F2C62
   25 6C08  6F6F7473
   25 6C0C  74726170
   25 6C10  20617420
   25 6C14  286E6F64
   25 6C18  65292000
   26 6C1C  61746368    wattxt: defm 'atch network events',0
   26 6C20  206E6574
   26 6C24  776F726B
   26 6C28  20657665
   26 6C2C  6E747300
   27 6C30  65742028    psttxt: defm 'et (link) ',0
   27 6C34  6C696E6B
   27 6C38  292000
   28 6C3B
```

```
                        ;Type hexadecimal word from BC.  Destroys AF.

   31 6C3B  78          h16out: ld a,b                  ;type high order byte
   32 6C3C  CD406C              call h8out
   33 6C3F  79                  ld a,c                  ; then low order...
                        ;       call h8out
                        ;       ret

                        ;Type hexadecimal byte from A.

   39 6C40  E5          h8out:  push hl                 ;preserve caller's HL.
   40 6C41  F5                  push af                 ;make a place on the sta
   41 6C42  210100              ld hl,1
   42 6C45  39                  add hl,sp               ;make HL point to A on
   43 6C46  CD4F6C              call hexc
   44 6C49  CD4F6C              call hexc
   45 6C4C  F1                  pop af                  ;restore character
   46 6C4D  E1                  pop hl                  ; and HL.
   47 6C4E  C9                  ret

                        ;Type hexadecimal digit from (HL).  Destroys AF and (HL)

   51 6C4F  AF          hexc:   xor a                   ;clear recipient byte.
   52 6C50  ED6F                rld                     ;rotate nyble from (HL)
   53 6C52  C630                add a,'0'               ;ASCIIfy A.
   54 6C54  FE3A                cp '9'+1                ;check for greater than
   55 6C56  3802                jr c,hex2
   56 6C58  C607                add a,'A'-'0'-10        ;adjust to A-F.
   57 6C5A  C3A05F      hex2:   jp chrout
   58 6C5D
```

```
                        ;Read some amount of hex into BC.  Return no carry if no
                        ;terminator in A.

 62 6C5D  010000        hexin:  ld bc,0                 ;clear recipient.
 63 6C60  CD7F6C                call chrinh              ;get first character.
 64 6C63  3806                  jr c,hexinf              ;success, proceed.
 65 6C65  C9                    ret                      ;return with carry not s
 66 6C66                                                 ;character in A as usua

 68 6C66  CD7F6C        hexinl: call chrinh              ;get a character.
 69 6C69  3C2D                  jr nc,chrihr             ;set carry and return.

 71 6C6B  CB21          hexinf: sla c                   ;multiply
 72 6C6D  CB10                  rl b                     ; previous
 73 6C6F  CB21                  sla c                    ; number
 74 6C71  CB10                  rl b                     ; by
 75 6C73  CB21                  sla c                    ; 16.
 76 6C75  CB10                  rl b                     ;...
 77 6C77  CB21                  sla c                    ;...
 78 6C79  CB10                  rl b                     ;...
 79 6C7B  B1                    or c                     ;accumulate new digit.
 80 6C7C  4F                    ld c,a
 81 6C7D  18E7                  jr hexinl                ;get more characters.

 83 6C7F  CDF65F        chrinh: call echin
 84 6C82  FE30                  cp '0'                   ;less than zero?
 85 6C84  3812                  jr c,chrihr              ;illegal hex; clear carr
 86 6C86  D630                  sub '0'
 87 6C88  FE0A                  cp 9+1                   ;more than 9?
 88 6C8A  D8                    ret c                    ;no, fine.
 89 6C8B  D611                  sub 'A'-'0'              ;convert A-F
 90 6C8D  FE06                  cp 6                     ;within bounds?
 91 6C8F  3C04                  jr nc,chrinb             ;bad character
 92 6C91  C60A                  add a,10                 ;fine character, convert
 93 6C93  37                    scf                      ; set carry,
 94 6C94  C9                    ret                      ; and return.

 96 6C95  C641          chrinb: add a,'A'                ;add back cruft to rest
 97 6C97  37                    scf                      ;set carry flag,

 99 6C98  3F            chrihr: ccf                     ;toggle carry flag
100 6C99  C9                    ret                      ; and return.
101 6C9A
```

```
                        ;Examine and deposit memory.

104 6C9A  20o6E6F64     alntxt: defm ' node ',0
104 6C9E  652000
105 6CA1  20494F50      ioptxt: defm ' IOP ',0
105 6CA5  2000
106 6CA7  61742000      whrtxt: defm 'at ',0
107 6CAB  636F6E74      acntxt: defm 'containing ',0
107 6CAF  61696E69
107 6CB3  6E672000
108 6CB7  20776974      wittxt: defm ' with ',0
108 6CBB  682000
109 6CBE  6E756C20      nulerr: defm 'nul argument illegal',0
109 6CC2  61726775
109 6CC6  6D656E74
109 6CCA  20696C6C
109 6CCE  6567616C
109 6CD2  00

111 6CD3  CD2863        altr:   call pvchek             ;must be privileged.
112 6CD6  219A6C                ld hl,alntxt
113 6CD9  CD9362                call strout
114 6CDC  CD156C                call decinc             ;get node number.
115 6CDF  CDF662                call whchek             ;check for white space
116 6CE2  CDA863                call bxchek             ; and valid box.
117 6CE5  41                    ld b,c                  ;convert to <node
118 6CE6  0E00                  ld c,0                  ; .0> and store
119 6CE8  C5                    push bc                 ; as destination.
120 6CE9  C5                    push bc                 ;(must use twice).

122 6CEA  21A16C                ld hl,ioptxt
123 6CED  CD9362                call strout
124 6CF0  CD5D6C                call hexin
125 6CF3  3802                  jr c,altiop             ;if typed something, bel
126 6CF5  0EFF                  ld c,0ffh               ;nil MPC.

128 6CF7  CDF662        altiop: call whchek             ;check for proper termin

130 6CFA  3EF0                  ld a,0f0h               ;OR in high nybble.
131 6CFC  B1                    or c
132 6CFD  5F                    ld e,a                  ;copy MPC address

134 6CFE  21A76C                ld hl,whrtxt            ;prompt for where to dep
135 6D01  CD9362                call strout
136 6D04  CD5D6C                call hexin              ;read the address.
137 6D07  21BE6C                ld hl,nulerr
138 6D0A  D25F59                jp nc,error
139 6D0D  CDF662                call whchek             ;check for white-space

141 6D10  C5                    push bc                 ;push address and put
142 6D11  E1                    pop hl                  ; it in a more useful AC
143 6D12  E3                    ex (sp),hl              ;put address on stack, o
144 6D13  E5                    push hl                 ;give <node.0> to RCALL
145 6D14  21486D                ld hl,xexmn             ;load lengthy routine to
146 6D17  E5                    push hl
147 6D18  CD0268                call rcall
148 6D1B  E2B05B                jp po,netgon            ;couldn't access node.
```

149 6D1E

```
 150 6D1E  F5              push af             ;save the byte.
 151 6D1F  21A86C          ld hl,acntxt
 152 6D22  CD9362          call strout         ;" containing "
 153 6D25  F1              pop af
 154 6D26  CD406C          call h8out          ;write byte.

                    ;Have displayed existing byte at the address, now get a

 158 6D29  21B76C          ld hl,wittxt        ;prompt for value.
 159 6D2C  CD9362          call strout

 161 6D2F  CD5D6C          call hexin          ;read the byte.
 162 6D32  3012            jr nc,nextp         ;if none, was just an ex

 164 6D34  51              ld d,c              ;copy the byte to D (E h
 165 6D35  C1              pop bc              ;restore address to exam
 166 6D36                                      ; <node.0> is still on
 167 6D36  CDBE62          call cnfrm
 168 6D39  21566D          ld hl,xdeps         ;remotely call deposit r
 169 6D3C  E5              push hl
 170 6D3D  CD0268          call rcall
 171 6D40  E2B058          jp po,netgon
 172 6D43  C36E59          jp next

 174 6D46  C1        nextp: pop bc             ;pop address to examine
 175 6D47  C1               pop bc             ;pop <node.0>
 176 6D48  C36E59           jp next            ;get next command.

 178 6D4B  7B        xexmn: ld a,e             ;get MPC address
 179 6D4C  D3FF             out (select),a     ;select it.
 180 6D4E  FDE3             ex (sp),iy
 181 6D50  FDE3             ex (sp),iy
 182 6D52  0A               ld a,(bc)          ;remote examine.
 183 6D53  C3A369           jp mpcuns          ;deselect MPC and return

 185 6D56  7B        xdeps: ld a,e             ;get MPC address.
 186 6D57  D3FF             out (select),a
 187 6D59  FDE3             ex (sp),iy
 188 6D5B  FDE3             ex (sp),iy
 189 6D5D  7A               ld a,d             ;remote deposit.
 190 6D5E  02               ld (bc),a
 191 6D5F  C3A369           jp mpcuns
 192 6D62
```

```
                        ;!Jump

 196 6D62  CD7D62       jump:   call boxin
 197 6D65  C5                   push bc
 198 6D66  2103F0               ld hl,rdie              ;do a CALL RDIE
 199 6D69  E5                   push hl
 200 6D6A  CD0268               call rcall              ;kill it
 201 6D6D  C36E59               jp next
 202 6D70
```

```
                        ;!Watch

  205 6D70  0D0A        whetxt: defb 13,10
  206 6D72  4576656E            defm 'Event   From    To   Missed',13,10,13,10,0
  206 6D76  74202020
  206 6D7A  46726F6D
  206 6D7E  20202020
  206 6D82  546F2020
  206 6D86  204D6973
  206 6D8A  73656400
  206 6D8E  0A0D0A00
  207 6D92  20434E43    wcotxt: defm ' CNCT ',0
  207 6D96  542000
  208 6D99  20444953    wdctxt: defm ' DISC ',0
  208 6D9D  432000
  209 6DA0  0D0A        wuptxt: defb 13,10
  210 6DA2  2A555020            defm '*UP   ',ctrlg,13,10,0
  210 6DA6  20200700
  210 6DAA  0A00
  211 6DAC  0D0A        wdntxt: defb 13,10
  212 6DAE  2A444F57            defm '*DOWN ',ctrlg,13,10,0
  212 6DB2  4E200700
  212 6DB6  0A00

  214 6DB8  CDC562      watch:  call cnfirm             ;get a confirmation
  215 6DBB  CD1264              call givlok             ;give up lock - watch t
  216 6DBE  21706D              ld hl,whetxt            ;type heading
  217 6DC1  CD9362              call strout
  218 6DC4  CD446E              call watrut             ;output routing info fi

  220 6DC7  3A8742      wattop: ld a,(newrut)
  221 6DCA  B7                  or a
  222 6DCB  C4446E              call nz,watrut          ;yes - watch it
  223 6DCE  010000              ld bc,0                 ;start at 0.0
  224 6DD1  CDEE6A      watlop: call mpckbc             ;any user characters?
  225 6DD4  C2846E              jp nz,watend            ;yes, terminate
  226 6DD7  CDE66D              call watnod             ;watch this node
  227 6DDA  04                  inc b                   ;step to next
  228 6DDB  78                  ld a,b
  229 6DDC  FE10                cp nbox
  230 6DDE  38F1                jr c,watlop             ;more to do
  231 6DE0  CD4255              call db1                ;wait a second
  232 6DE3  18E2                jr wattop               ;and do another
  233 6DE5
```

```
                        ;Subroutine to do the watch for one node (in BC).

 236 6DE5  C1           watnlp: pop bc                  ;get real destination ba
 237 6DE6  C5           watnod: push bc                 ;save BC on stack for la
 238 6DE7  C5                   push bc                 ;destination
 239 6DE8  219C6E               ld hl,xwatch            ;remote subr
 240 6DEB  E5                   push hl
 241 6DEC  CD0268               call rcall              ;poof
 242 6DEF  E22F6E               jp po,watret            ;if unreachable, just re

                        ;XWATCH returns A=LTYPE, C/NC, BC=link, DE=REMB/REML, H

 246 6DF2  3038                 jr nc,watret            ;if nothing to do there,
 247 6DF4  E5                   push hl                 ;save flags, LWATCH
 248 6DF5  CB67                 bit vbit,a              ;a VLINK?
 249 6DF7  200E                 jr nz,watv              ;yes, do it

 251 6DF9  CB7C         watth:  bit cnvbit,h            ;in or out of connection
 252 6DFB  2805                 jr z,watdsc             ;out - disconnect
 253 6DFD  21926D               ld hl,wcotxt            ;connection
 254 6E00  1811                 jr watstr
 255 6E02  21996D       watdsc: ld hl,wdctxt            ;disconnection
 256 6E05  180C                 jr watstr

 258 6E07  CB5C         watv:   bit vupbit,h            ;UP or DOWN?
 259 6E09  2805                 jr z,watdwn
 260 6E0B  21A06D               ld hl,wuptxt
 261 6E0E  1803                 jr watstr
 262 6E10  21AC6D       watdwn: ld hl,wdntxt

 264 6E13  CD9362       watstr: call strout             ;output string from HL
 265 6E16  CD5E62               call dstfou             ;formatted source outpu
 266 6E19  CD9E5F               call blank              ;a blank
 267 6E1C  42                   ld b,d
 268 6E1D  48                   ld c,e                  ;and destination
 269 6E1E  CD5E62               call dstfou             ;more
 270 6E21  CD9E5F               call blank
 271 6E24  C1                   pop bc                  ;get L into C (LWATCH)
 272 6E25  79                   ld a,c
 273 6E26  B7                   or a
 274 6E27  C46B6C               call nz,decout          ;else output it
 275 6E2A  CD9D62               call crlf
 276 6E2D  18B6                 jr watnlp

 278 6E2F  C1           watret: pop bc                  ;restore destination
 279 6E30  C9                   ret                     ;and return
 280 6E31
```

```
                        ;Subroutine to print out the routing table

 283 6E31  0D0A         wrttxt: defb 13,10
 284 6E33  56426F78             defm 'VBox Cost  Via',13,10,0
 284 6E37  20436F73
 284 6E3B  74202056
 284 6E3F  69610D0A
 284 6E43  00

 286 6E44  21316E       watrut: ld hl,wrttxt
 287 6E47  CD9362               call strout

 289 6E4A  110000               ld de,0                 ;start with node 0
 290 6E4D  21F842       wrtlop: ld hl,costs             ;get cost table
 291 6E50  19                   add hl,de
 292 6E51  7E                   ld a,(hl)               ;read it
 293 6E52  FEFF                 cp 0FFh
 294 6E54  2820                 jr z,wrtnxt             ;infinite - skip it
 295 6E56  48                   ld c,e                  ;get node number
 296 6E57  CD906C               call dec31              ;output it
 297 6E5A  CD9E5F               call blank              ;a space
 298 6E5D  CD9E5F               call blank              ;a second space
 299 6E60  4E                   ld c,(hl)               ;get cost back
 300 6E61  CD906C               call dec31
 301 6E64  CD9E5F               call blank
 302 6E67  01F0FF               ld bc,routes-costs
 303 6E6A  09                   add hl,bc
 304 6E6B  4E                   ld c,(hl)               ;pick out route
 305 6E6C  3A0042               ld a,(me)
 306 6E6F  47                   ld b,a                  ;fudge it to look like a
 307 6E70  CD5E62               call dstfou             ;output it
 308 6E73  CD9062               call crlf
 309 6E76  1C           wrtnxt: inc e                   ;next node, please
 310 6E77  3E10                 ld a,nbox               ;done them all?
 311 6E79  BB                   cp e
 312 6E7A  38D1                 jr c,wrtlop             ;no, do more
 313 6E7C  AF                   xor a
 314 6E7D  328742               ld (newrut),a           ;clear flag
 315 6E80  CD9062               call crlf
 316 6E83  C9                   ret                     ;else return

 318 6E84  CD5E59       watend: call errorp
 319 6E87  0D0A                 defb 13,10
 320 6E89  57617463             defm 'Watch terminated',13,10,0
 320 6E8D  68207465
 320 6E91  7260696E
 320 6E95  61746564
 320 6E99  0D0A00
 321 6E9C
```

```
                        ;Remote watch subroutine

 324 6E9C  FDE5         xwatch: push iy                 ;save caller's IY
 325 6E9E  FD211843             ld iy,lblk0             ;go through all the line

 327 6EA2  FD7E1A       xwatlp: ld a,(iy+lwatch)        ;anything to watch here?
 328 6EA5  B7                   or a
 329 6EA6  281D                 jr z,xwatnx             ;nope
 330 6EA8  6F                   ld l,a                  ;yes, store it in L
 331 6EA9  2D                   dec l                   ;less the one we're doin
 332 6EAA  FD361A00             ld (iy+lwatch),0        ;and clear the watch wo
 333 6EAE  3A0042               ld a,(me)               ;set up BC=this link
 334 6EB1  47                   ld b,a
 335 6EB2  FD4E0D               ld c,(iy+lnum)
 336 6EB5  FD5618               ld d,(iy+lremb)         ;and DE=remb/reml
 337 6EB8  FD5E19               ld e,(iy+lreml)
 338 6EBB  FD6614               ld h,(iy+lflag1)        ;and H=flags
 339 6EBE  FD7E0D               ld a,(iy+ltype1)        ;and A=LTYPE
 340 6EC1  37                   scf                     ;set carry
 341 6EC2  FDE1                 pop iy
 342 6EC4  C9                   ret

 344 6EC5  CDD664       xwatnx: call iynext             ;do next line
 345 6EC8  38D8                 jr c,xwatlp             ;untill all done
 346 6ECA  FDE1                 pop iy                  ;else return with NC (fr
 347 6ECC  C9                   ret
 348 6ECD
```

```
                        ;This code is responsible for permanently changing netwo
                        ;and writing them to disk.  In all cases, the permanent
                        ;the first part of the line block for the line in questi
                        ;changes have been made, the line is reset so the parame
                        ;effect.  Then, the new line block is written to disk.


  357 6ECD  CD0A62      pst:    call lnkin              ;get the desired destina
  358 6ED0  C5                  push bc                 ;save it on the stack fo

  360 6ED1  CD2D72              call getblk             ;get this line's LBLK in

  362 6ED4  21DA6E              ld hl,psttab            ;get ptr to command tabl
  363 6ED7  C3A359              jp disptc               ;and process it

  365 6EDA  41          psttab: defb 'A'
  366 6EDB  7A6F1C6F            defw aut,auttxt
  367 6EDF  43                  defb 'C'
  368 6EE0  866F286F            defw clp,clptxt
  369 6EE4  44                  defb 'D'
  370 6EE5  976F3A6F            defw dly,dlytxt
  371 6EE9  45                  defb 'E'
  372 6EEA  A66F4E5E            defw eco,ecotxt
  373 6EEE  46                  defb 'F'
  374 6EEF  B76F375E            defw flo,flotxt
  375 6EF3  47                  defb 'G'
  376 6EF4  0270486F            defw grp,grptxt
  377 6EF8  48                  defb 'H'
  378 6EF9  11704F6F            defw han,hantxt
  379 6EFD  49                  defb 'I'
  380 6EFE  5670445E            defw int,inttxt
  381 6F02  4C                  defb 'L'
  382 6F03  B2705B6F            defw lik,liktxt
  383 6F07  50                  defb 'P'
  384 6F08  B870606F            defw par,partxt
  385 6F0C  53                  defb 'S'
  386 6F0D  E670676F            defw spd,spdtxt
  387 6F11  54                  defb 'T'
  388 6F12  EF706D6F            defw typ,typtxt
  389 6F16  55                  defb 'U'
  390 6F17  6271726F            defw upd,updtxt
  391 6F1B  FF                  defb -1

  393 6F1C  75746F63    auttxt: defm 'utoconnect ',0
  393 6F20  6F6E6E65
  393 6F24  63742000
  394 6F28  6C697020    clptxt: defm 'lip for XON/XOFF ',0
  394 6F2C  666F7220
  394 6F30  584F4E2F
  394 6F34  584F4646
  394 6F38  2000
  395 6F3A  656C6179    dlytxt: defm 'elay/padding ',0
  395 6F3E  2F706164
  395 6F42  64696E67
  395 6F46  2000

                        ;ecotxt: defl ecotxt                    ;see V.Z80
```

```
 397 5E37                flotxt: defl sfltxt             ;see V.Z80
 398 6F48  726F7570      grptxt: defm 'roups ',0
 398 6F4C  732000
 399 6F4F  616E6473      hantxt: defm 'andshaking ',0
 399 6F53  68616869
 399 6F57  6E672000
 400 5E44                inttxt: defl sintxt             ;see V.Z80
 401 6F5B  69686520      liktxt: defm 'ike ',0
 401 6F5F  00
 402 6F60  61726974      partxt: defm 'arity ',0
 402 6F64  792000
 403 6F67  70656564      spdtxt: defm 'peed ',0
 403 6F6B  2000
 404 6F6D  79706520      typtxt: defm 'ype ',0
 404 6F71  00
 405 6F72  70646174      updtxt: defm 'pdated ',0
 405 6F76  65642000
 406 6F7A
```

```
                        ;!Set Autoconnect

409 6F7A  CDEC61        aut:    call dstin              ;get the destination
410 6F7D  DD7005                ld (ix+lautoh),b
411 6F80  DD7106                ld (ix+lautol),c
412 6F83  C38A71                jp thonly


                        ;!Set Clip

417 6F86  CD156C        clp:    call decinc             ;get clip value
418 6F89  79                    ld a,c
419 6F8A  B7                    or a
420 6F8B  CA2A6C                jp z,decine
421 6F8E  FA2A6C                jp m,decine
422 6F91  DD7703                ld (ix+lclip),a
423 6F94  C39671                jp xsetdo


                        ;!Set Delay/Padding

428 6F97  CD156C        dly:    call decinc             ;get padding value
429 6F9A  79                    ld a,c
430 6F9B  FE10                  cp 16
431 6F9D  D22A6C                jp nc,decine
432 6FA0  DD7708                ld (ix+lpad),a
433 6FA3  C38A71                jp thonly


                        ;!Set Echo

438 6FA6  CDDE62        eco:    call yesno              ;get an answer
439 6FA9  DDCB0CAE              res echbit,(ix+ltype1)  ;assume no-echo
440 6FAD  CA8A71                jp z,thonly             ;if NO, guessed right
441 6FB0  DDCB0CEE              set echbit,(ix+ltype1)
442 6FB4  C38A71                jp thonly


                        ;!Set Flow Control

447 6FB7  DDCB01B6      flo:    res pfxbit,(ix+ltype2)  ;assume neither
448 6FBB  DDCB01BE              res gfxbit,(ix+ltype2)
449 6FBF  CDA259                call dispth             ;in-line table
450 6FC2  50                    defb 'P'
451 6FC3  FB6FD76F              defw flp,flptxt
452 6FC7  47                    defb 'G'
453 6FC8  FC6FDD6F              defw flg,flgtxt
454 6FCC  42                    defb 'B'
455 6FCD  F76FE56F              defw flb,flbtxt
456 6FD1  4E                    defb 'N'
457 6FD2  FF6FE96F              defw fln,flntxt
458 6FD6  FF                    defb -1

460 6FD7  726F6365      flptxt: defm 'roces',0
460 6FDB  7300
461 6FDD  656E6572      flgtxt: defm 'enerate',0
461 6FE1  61746500
```

```
462 6FE5  6F746800    flbtxt: defm 'oth',0
463 6FE9  65697468    flntxt: defm 'either',0
463 6FED  657200
465 6FF0  DDCB01FE    flg:    set gfxbit,(ix+ltype2)
466 6FF4  C38A71              jp thonly
468 6FF7  DDCB01FE    flb:    set gfxbit,(ix+ltype2)
469 6FFB  DDCB01F6    flp:    set pfxbit,(ix+ltype2)
470 6FFF  C38A71      fln:    jp thonly

                      ;!Set Groups originate answer

475 7002  CD8761      grp:    call grpin              ;get a group
476 7005  DD7107              ld (ix+logrp),c
477 7008  CD8761              call grpin
478 700B  DD7108              ld (ix+lagrp),c
479 700E  C38A71              jp thonly

                      ;!Set Handshaking

483 7011  DDCB008E    han:    res shbit,(ix+ltype1)   ;assume simple DCE
484 7015  DDCB00C6            set dcebit,(ix+ltype1)
485 7019  CDA259              call dispth
486 701C  43                  defb 'C'
487 701D  4C703170            defw hnc,hnctxt
488 7021  54                  defb 'T'
489 7022  48703670            defw hnt,hnttxt
490 7026  44                  defb 'D'
491 7027  4F703870            defw hnd,hndtxt
492 702B  53                  defb 'S'
493 702C  53704270            defw hns,hnstxt
494 7030  FF                  defb -1

496 7031  3D444345    hnctxt: defm '=DCE',0
496 7035  00
497 7036  3D445445    hnttxt: defm '=DTE',0
497 703A  00
498 703B  61746173    hndtxt: defm 'ataset',0
498 703F  657400
499 7042  696D706C    hnstxt: defm 'imple',0
499 7046  6500

501 7048  DDCB0086    hnt:    res dcebit,(ix+ltype1)
502 704C  C39671      hnc:    jp xsetdo
503 704F  DDCB00CE    hnd:    set shbit,(ix+ltype1)
504 7053  C38A71      hns:    jp thonly

                      ;!Set Interrupt

509 7056  DDCB00B6    int:    res pbkbit,(ix+ltype1)  ;assume break
510 705A  DD3600C0            ld (ix+lintr),0         ;and no interrupt
511 705E  CDA259              call dispth
512 7061  43                  defb 'C'
513 7062  8C707170            defw itc,itctxt
514 7066  42                  defb 'B'
```

```
515 7067  AF708370          defw itb,itbtxt
516 706B  4E                defb 'N'
517 706C  AB708870          defw itn,itntxt
518 7070  FF                defb -1

520 7071  68617261  itctxt: defm 'haracter (enter) ',0
520 7075  63746572
520 7079  2028656E
520 707D  74657229
520 7081  2000
521 7083  72656168  itbtxt: defm 'reak',0
521 7087  00
522 7088  6F6E6500  itntxt: defm 'one',0

524 708C  CD925F    itc:    call imgin          ;get image mode charact
525 708F  F5                push af             ;save it
526 7090  CD8A62            call stroth
527 7093  20287265          defm ' (re-enter) ',0
527 7097  2D656E74
527 709B  65722920
527 709F  00
528 70A0  CD925F            call imgin          ;get another
529 70A3  C1                pop bc              ;get first into A, secon
530 70A4  B8                cp b                ;a match?
531 70A5  C2FC62            jp nz,noncnf        ;no, say not confirmed
532 70A8  DD770C            ld (ix+lintr),a
533 70AB  DDCB0CF6  itn:    set pbkbit,(ix+ltype1)
534 70AF  C38A71    itb:    jp thonly

                    ;!Set Like

539 70B2  CD0A62    lik:    call lnkin          ;get other destination
540 70B5  CD2D72            call getblk         ;read its line block ins
541 70B8  C39671            jp xsetdo           ;and then do this one li

                    ;!Set Parity

546 70BB  DD360900  par:    ld (ix+lprty),0     ;assume none
547 70BF  CDA259            call dispth
548 70C2  45                defb 'E'
549 70C3  DD700270          defw pae,paetxt
550 70C7  4F                defb 'O'
551 70C8  EC70D670          defw pao,paotxt
552 70CC  4E                defb 'N'
553 70CD  E370D970          defw pan,pantxt
554 70D1  FF                defb -1

556 70D2  76656E00  paetxt: defm 'ven',0
557 70D6  646400    paotxt: defm 'dd',0
558 70D9  6F6E6500  pantxt: defm 'one',0

560 70DD  DD3409    pae:    inc (ix+lprty)      ; 2 = even
561 70E0  DD3409    pao:    inc (ix+lprty)      ; 1 = odd
562 70E3  C38A71    pan:    jp thonly           ; 0 = none
```

```
                        ;!Set Speed

 567 70E6  CDA66C       spd:    call ratin              ;get the desired rate
 568 70E9  DD770A               ld (ix+lrate),a         ;update it
 569 70EC  C39671               jp xsetdo


                        ;!Set Type

 574 70EF  DDCB0C9E     typ:    res onbit,(ix+ltype1)   ;assume None
 575 70F3  DDCB0CA6             res vbit,(ix+ltype1)
 576 70F7  DDCB0C96             res hbit,(ix+ltype1)
 577 70FB  CDA259               call dispth
 578 70FE  56                   defb 'V'
 579 70FF  45721371             defw tyv,tyvtxt
 580 7103  54                   defb 'T'
 581 7104  55711C71             defw tyt,tyttxt
 582 7108  48                   defb 'H'
 583 7109  51713171             defw tyh,tyhtxt
 584 710D  4E                   defb 'N'
 585 710E  5F714271             defw tyn,tyntxt
 586 7112  FF                   defb -1

 588 7113  626F7820   tyvtxt: defm 'box Link',0
 588 7117  4C696E6B
 588 711B  00
 589 711C  65726D69   tyttxt: defm 'erminal Link (area) ',0
 589 7120  6E616C20
 589 7124  4C696E6B
 589 7128  20286172
 589 712C  65612920
 589 7130  00
 590 7131  6F737420   tyhtxt: defm 'ost Link (host) ',0
 590 7135  4C696E6B
 590 7139  2028686F
 590 713D  73742920
 590 7141  00
 591 7142  6F742063   tyntxt: defm 'ot configured ',0
 591 7146  6F6E6669
 591 714A  67757265
 591 714E  642000

 593 7151  DDCB0CD6   tyh:    set hbit,(ix+ltype1)
 594 7155  CD156C     tyt:    call decinc             ;get host/area
 595 7158  DD7104             ld (ix+lhost),c
 596 715B  DDCB0CDE           set onbit,(ix+ltype1)
 597 715F  C39671     tyn:    jp xsetdo

 599 7162  C39671     upd:    jp xsetdo               ;updated is basically a
 600 7165
```

```
                        ;XSETDO will copy the DBLK to the destination AND to the
                        ;net address of the affected link is on the stack.  Ent
                        ;to first make sure it's a T or H link.

605 001C                pstlen: equ prclen+dsize        ;size of remote set DBL

607 7165  CD5E59        therr:  call errorp
608 7168  3F787878              defm '?xxx Illegal except for T/H Links',0
608 716C  20496C6C
608 7170  6567616C
608 7174  20657863
608 7178  65707420
608 717C  666F7220
608 7180  542F4820
608 7184  4C696E68
608 7188  7300

610 718A  DDCB005E      thonly: bit onbit,(ix+ltype1)   ;must be on-line
611 718E  28D5                  jr z,therr
612 7190  DDCB0066              bit vbit,(ix+ltype1)
613 7194  20CF                  jr nz,therr

615 7196  CDC562        xsetdo: call cnfirm             ;get confirmation
616 7199  C1                    pop bc                 ;get destination off sta
617 719A  C5                    push bc                ;and save it again
618 719B  C5                    push bc                ;and give it to RCALL,
619 719C  CD2D72                call setblk            ;update the disk

                        ;Now do the long RCALL to the destination.  Note that w
                        ;Trash IX because we can re-compute it again.  The desti
                        ;on the stack twice - once for RCALL, the other to save

625 719F  CDDA68                call getbuf            ;get a buffer
626 71A2  FDCB14EE              set lrcbit,(iy+lflag1) ;long RCALL
627 71A6  DD36051C              ld (ix+plen),pstlen    ;for sets
628 71AA  214272                ld hl,xsetit           ;subroutine address
629 71AD  E5                    push hl                ;save that, too

                        ;Compute the address within the RCALL to get the DBLK in

633 71AE  DDE5                  push ix
634 71B0  E1                    pop hl                 ;copy the packet pointer
635 71B1  111400                ld de,plroff           ;add in the offset for l
636 71B4  19                    add hl,de
637 71B5  EB                    ex de,hl               ;put destination into DE
638 71B6  21D068                ld hl,dblk             ;get source      into HL
639 71B9  010D00                ld bc,dsize            ;get size        into BC
640 71BC  EDB0                  ldir                   ;copy the DBLK into the
641 71BE  CD0268                call rcall             ;do the RCALL

                        ;Now tell the user whether we updated the network or not

645 71C1  FDCB14AE              res lrcbit,(iy+lflag1)
646 71C5  21F471                ld hl,snntxt           ;assume not in network
647 71C8  E25F59                jp po,error            ;if RCALL lost, buffer g
648 71CB  CD0269                call givbuf            ;else give up the buffe
649 71CE  CD5E59                call errorp
```

```
650 71D1  44697368          defm 'Disk and network update complete',13,10,0
650 71D5  20616E64
650 71D9  206E6574
650 71DD  776F726B
650 71E1  20757064
650 71E5  61746520
650 71E9  636F6D70
650 71ED  6C657465
650 71F1  0D0A00
651 71F4  5761726E  snntxt: defm 'Warning: Update sucessful for disk but not
651 71F8  696E673A
651 71FC  20557064
651 7200  61746520
651 7204  73756363
651 7208  73736675
651 720C  6C20666F
651 7210  72206469
651 7214  736B2062
651 7218  75742C6E
651 721C  6F742C66
651 7220  6F72206E
651 7224  6574776F
651 7228  726B0D0A
651 722C  00
652 722D
```

```
  653 722D                getblk:
  654 722D  CD5E59        setblk: call errorp
  655 7230  4E6F2044              defm 'No Disk - aborted',0
  655 7234  69736B20
  655 7238  2D206162
  655 723C  6F727465
  655 7240  6400

  657 7242  CD03FC        xsetit: call rdie
  658 7245  CD03FC        tyv:    call rdie
 4179 7248
```

```
4180 7248                patch:  defs vpat               ;leave patch space

4183 7288                vbuf0:  equ $

   0 7288                *include cnf.z80
   1 7288                cname:  macro #nn
   2 7288                        org name
   3 7288                        defm #nn,0
   4 7288                        endm

   6 7288                cnode:  macro #nn
   7 7288                ff:     defl (#nn.eq.nme)
   8 7288                        if (#nn.gt.nbox)
   9 7288                error*  cnode bigger than nbox
  10 7288                        endif
  11 7288                mymsk:  defl 1                  ;zero'th line has this
  12 7288                        endm

  14 7288                clink:  macro #nn
  15 7288                ll:     defl #nn
  16 7288                        cdefau
  17 7288                        endm

  19 7288                cevpar: macro
  20 7288                cprty:  defl 2
  21 7288                        endm

  23 7288                cauto:  macro #dd1,#dd2
  24 7288                cautoh: defl #dd1
  25 7288                cautol: defl #dd2
  26 7288                cflow:  defl cflow.or.autmsk
  27 7288                        endm

  29 7288                crate:  macro #rr
  30 7288                cirate: defl b#rr
  31 7288                        endm

  33 7288                cvlink: macro
  34 7288                ctype:  defl (ctype.and.(.not.hmsk)).or.onmsk.or.vmsk
  35 7288                        endm

  37 7288                cdte:   macro
  38 7288                ctype:  defl (ctype.and.(.not.dcemsk))
  39 7288                        endm

  41 7288                cdce:   macro
  42 7288                ctype:  defl (ctype.or.dcemsk)
  43 7288                        endm

  45 7288                chand:  macro #hh
  46 7288                ctype:  defl (ctype.and.(.not.shmsk)).or.(#hh*shmsk)
  47 7288                        endm
  48 7288
```

```
 49 7288                chlink: macro #hh
 50 7288                chost:  defl #hh
 51 7288                ctype:  defl (ctype.and.(.not.vmsk)).or.onmsk.or.hmsk
 52 7288                        endm

 54 7288                ctlink: macro #hh
 55 7288                chost:  defl #hh
 56 7288                ctype:  defl (ctype.and.(.not.(vmsk.or.hmsk))).or.onmsk
 57 7288                        endm

 59 7288                cecho:  macro
 60 7288                ctype:  defl (ctype.or.echmsk)
 61 7288                        endm

 63 7288                cpflow: macro
 64 7288                cflow:  defl (cflow.or.pfxmsk)
 65 7288                        endm

 67 7288                cgflow: macro #cc
 68 7288                cflow:  defl (cflow.or.gfxmsk)
 69 7288                ciclip: defl #cc
 70 7288                        endm

 72 7288                crflow: macro
 73 7288                cflow:  defl (cflow.or.prsmsk)
 74 7288                        endm

 76 7288                cogroup:        macro #gg
 77 7288                cogrp:  defl #gg
 78 7288                        endm

 80 7288                cagroup:        macro #gg
 81 7288                cagrp:  defl #gg
 82 7288                        endm

 84 7288                cdelay: macro #pp
 85 7288                cpad:   defl #pp
 86 7288                        endm

 88 7288                cinterrupt:     macro #ii
 89 7288                cintr:  defl #ii
 90 7288                ctype:  defl ctype.or.pbkmsk
 91 7288                        endm

 93 7288                cbreak: macro
 94 7288                cintr:  defl 0
 95 7288                ctype:  defl ctype.and.(.not.pbkmsk)
 96 7288                        endm

 98 7288                lblk:   macro #lin,#off,#val
 99 7288                        org lblk0+#lin*lsize+#off
100 7288                        defb #val
101 7288                        endm

103 7288                cend:   macro
104 7288                        if (ff)
105 7288                        lblk ll,ltype1,ctype
```

```
  106 7288                      lblk ll,ltype2,cflow
  107 7288                      lblk ll,lclip,ciclip
  108 7288                      lblk ll,lnum,ll
  109 7288                      lblk ll,lhost,chost
  110 7288                      lblk ll,lautoh,cautoh
  111 7288                      lblk ll,lautol,cautol
  112 7288                      lblk ll,logrp,cogrp
  113 7288                      lblk ll,lagrp,cagrp
  114 7288                      lblk ll,lmymsk,mymsk
  115 7288              mymsk:  defl mymsk*2
  116 7288                      if (mymsk.and.16)
  117 7288              mymsk:  defl 1
  118 7288                      endif
  119 7288                      lblk ll,lprty,cprty
  120 7288                      lblk ll,lrate,cirate
  121 7288                      lblk ll,lflag1,0
  122 7288                      lblk ll,lflag2,0
  123 7288                      lblk ll,ldbtim,0
  124 7288                      lblk ll,lcid,0
  125 7288                      lblk ll,lremb,0
  126 7288                      lblk ll,lreml,0
  127 7288                      lblk ll,lpad,cpad
  128 7288                      lblk ll,lintr,cintr
  129 7288                      endif
  130 7288                      endm

  132 7288              cdefau: macro
  133 7288              chost:  defl 0FFh
  134 7288              cautoh: defl 0FFh
  135 7288              cautol: defl 0FFh
  136 7288              cflow:  defl 0
  137 7288              cogrp:  defl 0
  138 7288              cagrp:  defl 0
  139 7288              ctype:  defl dcemsk             ;dead line, assume DCE t
  140 7288              cprty:  defl 0                  ;8 bit byte size, no par
  141 7288              cirate: defl bhunt
  142 7288              ciclip: defl 32
  143 7288              cpad:   defl 0
  144 7288              cintr:  defl 0
  145 7288                      endm
  146 7288
```

```
148 7288                bbntty: macro #ll,#ww,#ss
149 7288                                clink #ll
150 7288                                        cdce
151 7288                                        chand 0
152 7288                                        cbreak
153 7288                                        ctlink #ww
154 7288                                        cogroup g7msk
155 7288                                        cauto 255,0
156 7288                                        crate #ss
157 7288                                        cpflow
158 7288                                        cgflow 32
159 7288                                        if (b#ss.eq.b300)
160 7288                                        cdelay 10
161 7288                                        endif
162 7288                                cend
163 7288                        endm

165 7288                bbnhst: macro #ll,#ww
166 7288                                clink #ll
167 7288                                        cdce
168 7288                                        chand 0
169 7288                                        chlink #ww
170 7288                                        cogroup g7msk
171 7288                                        cauto 255,0
172 7288                                        crate 9600
173 7288                                        cpflow
174 7288                                        cgflow 32
175 7288                                cend
176 7288                        endm

178 7288                bbnvl:  macro #ll
179 7288                                clink #ll
180 7288                                        cdce
181 7288                                        chand 0
182 7288                                        cvlink
183 7288                                        crate 9600
184 7288                                        cpflow
185 7288                                        cgflow 32
186 7288                                cend
187 7288                        endm

189 7288                bbnoff: macro #ll
190 7288                                clink #ll
191 7288                                cend
192 7288                        endm

                        ;++CNF

196 7288                        cname 'BBN Development VTN'

                        ;               H o s t     N u m b e r s
                        ;
                        ;
                        ; Num           Host Name               VTN Addresses
                        ;
                        ;   0           Unassigned
```

```
;    1      BBNQ            0.20
;    2      BBNS            0.21
;    3      BBNT            0.22, 0.28
;    4      BBNW            0.23
;    5      BBNX            0.24, 0.26, 0.30
;    6      BBNV            0.25
;    7      BBNCD           0.27
;    8      BBNA            0.29 * Configured as a termina
;    9      NCC-TAC         0.31

;Area   What
;
; 0     PP
; 1     Hardware Lab
; 2     CDLab
; 3     Unassigned location
```

221 421A

```
223 421A
224 421A
225 421A
226 421A
227 421A
228 421A
229 421A
230 421A
231 421A
232 421A
233 421A
234 421A
235 421A
236 421A
237 421A
238 421A
239 421A
240 421A
241 421A
242 421A
243 421A
244 421A
245 421A
246 421A
247 421A
248 421A
249 421A
250 421A
251 421A
252 421A
253 421A
254 421A
255 421A

257 421A
```

```
cnode 0                     ;PP0, 32 ports
 bbnvl  0                   ;trunk to HL1 1.0
 bbntty 1,1,9600            ;VT52 in HL
 bbntty 2,3,9600            ;unused
 bbntty 3,3,hunt            ;unused
 bbnvl  4                   ;trunk to CD2 2.0
 bbntty 5,0,300             ;Console TI near PP
 bbntty 6,3,9600            ;unused
 bbntty 7,3,9600            ;unused
 bbnvl  8                   ;trunk to CD3 3.0
 bbntty 9,3,9600            ;unused
 bbntty 10,3,9600           ;unused
 bbntty 11,3,9600           ;unused
 bbntty 12,0,9600           ;patch panel 1
 bbntty 13,0,9600           ;patch panel 2
 bbntty 14,0,9600           ;patch panel 3
 bbntty 15,0,9600           ;patch panel 4
 bbntty 16,0,9600           ;patch panel 5
 bbntty 17,0,9600           ;patch panel 6
 bbntty 18,0,9600           ;patch panel 7
 bbntty 19,0,9600           ;patch panel 8
 bbnhst 20,1                ;patch panel 13 BBNQ
 bbnhst 21,2                ;patch panel 14 BBNS
 bbnhst 22,3                ;patch panel 15 BBNT
 bbnhst 23,4                ;patch panel 16 BBNW
 bbnhst 24,5                ;patch panel 17 BBNX
 bbnhst 25,6                ;patch panel 18 BBN-VAX
 bbnhst 26,5                ;patch panel 19 BBNX
 bbnhst 27,7                ;patch panel 20 BBN-CD (
 bbnhst 28,3                ;patch panel 21 BBNT
 bbntty 29,8,9600           ;patch panel 22 BBNA,
 bbnhst 30,5                ;patch panel 23 BBNX
 bbnhst 31,9                ;patch panel 24 NCC-TAC
```

```
 259 421A                cnode 1                 ;HL1, 8 ports
 260 421A                 bbnvl  0               ;trunk to PPO 0.0
 261 421A                 bbntty 1,3,9600        ;unused
 262 421A                 bbntty 2,3,9600        ;unused
 263 421A                 bbntty 3,3,hunt        ;unused
 264 421A                 bbntty 4,1,9600        ;random HL terminal
 265 421A                 bbntty 5,3,9600        ;unused
 266 421A                 bbntty 6,3,9600        ;unused
 267 421A                 bbntty 7,3,9600        ;unused
 268 421A                 bbnoff 8               ;non-existant
 269 421A                 bbnoff 9               ;non-existant
 270 421A                 bbnoff 10              ;non-existant
 271 421A                 bbnoff 11              ;non-existant
 272 421A                 bbnoff 12              ;non-existant
 273 421A                 bbnoff 13              ;non-existant
 274 421A                 bbnoff 14              ;non-existant
 275 421A                 bbnoff 15              ;non-existant
 276 421A                 bbnoff 16              ;non-existant
 277 421A                 bbnoff 17              ;non-existant
 278 421A                 bbnoff 18              ;non-existant
 279 421A                 bbnoff 19              ;non-existant
 280 421A                 bbnoff 20              ;non-existant
 281 421A                 bbnoff 21              ;non-existant
 282 421A                 bbnoff 22              ;non-existant
 283 421A                 bbnoff 23              ;non-existant
 284 421A                 bbnoff 24              ;non-existant
 285 421A                 bbnoff 25              ;non-existant
 286 421A                 bbnoff 26              ;non-existant
 287 421A                 bbnoff 27              ;non-existant
 288 421A                 bbnoff 28              ;non-existant
 289 421A                 bbnoff 29              ;non-existant
 290 421A                 bbnoff 30              ;non-existant
 291 421A                 bbnoff 31              ;non-existant

 293 421A
```

```
295 421A
296 421A                cnode 2                 ;CD2, 16 ports
297 421A                 bbnvl  0               ;trunk to PPO 0.4
298 421A                 bbntty 1,2,9600        ;CD VT100
299 421A                 bbntty 2,3,9600        ;unused
300 421A                 bbntty 3,3,hunt        ;unused
301 421A                 bbnvl  4               ;trunk to CD4 4.2
302 421A                 bbntty 5,3,9600        ;unused
303 421A                 bbntty 6,3,9600        ;unused
304 421A                 bbntty 7,3,9600        ;unused
305 421A                 bbnvl  8               ;trunk to CD3 3.4
306 421A                 bbntty 9,3,9600        ;unused
307 421A                 bbntty 10,3,9600       ;unused
308 421A                 bbntty 11,3,9600       ;unused
309 421A                 bbntty 12,2,300        ;CD TI terminal
310 421A                 bbntty 13,3,9600       ;unused
311 421A                 bbntty 14,3,9600       ;unused
312 421A                 bbntty 15,3,9600       ;unused
313 421A                 bbnoff 16              ;non-existant
314 421A                 bbnoff 17              ;non-existant
315 421A                 bbnoff 18              ;non-existant
316 421A                 bbnoff 19              ;non-existant
317 421A                 bbnoff 20              ;non-existant
318 421A                 bbnoff 21              ;non-existant
319 421A                 bbnoff 22              ;non-existant
320 421A                 bbnoff 23              ;non-existant
321 421A                 bbnoff 24              ;non-existant
322 421A                 bbnoff 25              ;non-existant
323 421A                 bbnoff 26              ;non-existant
324 421A                 bbnoff 27              ;non-existant
325 421A                 bbnoff 28              ;non-existant
326 421A                 bbnoff 29              ;non-existant
327 421A                 bbnoff 30              ;non-existant
                         bbnoff 31              ;non-existant

329 421A
```

```
331 421A                cnode 3                 ;CD3, 8 ports
332 421A                 bbnvl  0               ;trunk to PP0 0.8
333 421A                 bbnvl  1               ;trunk to CD4 4.1
334 421A                 bbntty 2,2,9600        ;CD Concept-100 Termina
335 421A                 bbntty 3,3,hunt        ;unused
336 421A                 bbnvl  4               ;trunk to CD2 2.8
337 421A                 bbntty 5,3,9600        ;unused
338 421A                 bbntty 6,3,9600        ;unused
339 421A                 bbntty 7,3,9600        ;unused
340 421A                 bbnoff 8               ;non-existant
341 421A                 bbnoff 9               ;non-existant
342 421A                 bbnoff 10              ;non-existant
343 421A                 bbnoff 11              ;non-existant
344 421A                 bbnoff 12              ;non-existant
345 421A                 bbnoff 13              ;non-existant
346 421A                 bbnoff 14              ;non-existant
347 421A                 bbnoff 15              ;non-existant
348 421A                 bbnoff 16              ;non-existant
349 421A                 bbnoff 17              ;non-existant
350 421A                 bbnoff 18              ;non-existant
351 421A                 bbnoff 19              ;non-existant
352 421A                 bbnoff 20              ;non-existant
353 421A                 bbnoff 21              ;non-existant
354 421A                 bbnoff 22              ;non-existant
355 421A                 bbnoff 23              ;non-existant
356 421A                 bbnoff 24              ;non-existant
357 421A                 bbnoff 25              ;non-existant
358 421A                 bbnoff 26              ;non-existant
359 421A                 bbnoff 27              ;non-existant
360 421A                 bbnoff 28              ;non-existant
361 421A                 bbnoff 29              ;non-existant
362 421A                 bbnoff 30              ;non-existant
363 421A                 bbnoff 31              ;non-existant

365 421A
```

```
 367 421A                cnode 4                 ;CD4, 4 ports
 368 421A                 bbnvl  0               ;trunk to PP0 0.12
 369 421A                 bbnvl  1               ;trunk to CD3 3.1
 370 421A                 bbnvl  2               ;trunk to CD2 2.4
 371 421A                 bbntty 3,3,hunt        ;unused
 372 421A                 bbnoff 4               ;non-existant
 373 421A                 bbnoff 5               ;non-existant
 374 421A                 bbnoff 6               ;non-existant
 375 421A                 bbnoff 7               ;non-existant
 376 421A                 bbnoff 8               ;non-existant
 377 421A                 bbnoff 9               ;non-existant
 378 421A                 bbnoff 10              ;non-existant
 379 421A                 bbnoff 11              ;non-existant
 380 421A                 bbnoff 12              ;non-existant
 381 421A                 bbnoff 13              ;non-existant
 382 421A                 bbnoff 14              ;non-existant
 383 421A                 bbnoff 15              ;non-existant
 384 421A                 bbnoff 16              ;non-existant
 385 421A                 bbnoff 17              ;non-existant
 386 421A                 bbnoff 18              ;non-existant
 387 421A                 bbnoff 19              ;non-existant
 388 421A                 bbnoff 20              ;non-existant
 389 421A                 bbnoff 21              ;non-existant
 390 421A                 bbnoff 22              ;non-existant
 391 421A                 bbnoff 23              ;non-existant
 392 421A                 bbnoff 24              ;non-existant
 393 421A                 bbnoff 25              ;non-existant
 394 421A                 bbnoff 26              ;non-existant
 395 421A                 bbnoff 27              ;non-existant
 396 421A                 bbnoff 28              ;non-existant
 397 421A                 bbnoff 29              ;non-existant
 398 421A                 bbnoff 30              ;non-existant
 399 421A                 bbnoff 31              ;non-existant

 401 421A
```

```
 403 421A                cnode 15                ;TEST15, 4 ports
 404 421A                 bbntty 0,3,9600
 405 4328                 bbntty 1,3,9600
 406 43A8                 bbntty 2,3,300
 407 4428                 bbnvl  3
 408 44A8                 bbnoff 4                ;non-existant
 409 4528                 bbnoff 5                ;non-existant
 410 45A8                 bbnoff 6                ;non-existant
 411 4628                 bbnoff 7                ;non-existant
 412 46A8                 bbnoff 8                ;non-existant
 413 4728                 bbnoff 9                ;non-existant
 414 47A8                 bbnoff 10               ;non-existant
 415 4828                 bbnoff 11               ;non-existant
 416 48A8                 bbnoff 12               ;non-existant
 417 4928                 bbnoff 13               ;non-existant
 418 49A8                 bbnoff 14               ;non-existant
 419 4A28                 bbnoff 15               ;non-existant
 420 4AA8                 bbnoff 16               ;non-existant
 421 4B28                 bbnoff 17               ;non-existant
 422 4BA8                 bbnoff 18               ;non-existant
 423 4C28                 bbnoff 19               ;non-existant
 424 4CA8                 bbnoff 20               ;non-existant
 425 4D28                 bbnoff 21               ;non-existant
 426 4DA8                 bbnoff 22               ;non-existant
 427 4E28                 bbnoff 23               ;non-existant
 428 4EA8                 bbnoff 24               ;non-existant
 429 4F28                 bbnoff 25               ;non-existant
 430 4FA8                 bbnoff 26               ;non-existant
 431 5028                 bbnoff 27               ;non-existant
 432 50A8                 bbnoff 28               ;non-existant
 433 5128                 bbnoff 29               ;non-existant
 434 51A8                 bbnoff 30               ;non-existant
 435 5228                 bbnoff 31               ;non-existant

   9 52A8                end
```

bbnhst		Macro	b4800	S	000C	cnc		5B32	db		553E
bbnoff		Macro	b50	S	0000	cncadr		5B60	db01		554C
bbntty		Macro	b600	S	0006	cncdo		5B33	db1		5542
bbnvl		Macro	b7200	S	000D	cncdsc		5B58	dblk		68D0
cagroup	S	Macro	b75	S	0001	cnchek		634F	dbmax		0003
cauto		Macro	b9600		000E	cncho		5BD3	dbmin		0001
cbreak		Macro	bhunt		00FF	cncl0		5BE0	dbr		5547
cdce		Macro	blank		5F9E	cncl1		5BE8	dcebit		0000
cdefau		Macro	blktab		42C8	cncl2		5BF4	dcemsk		0001
cdelay		Macro	boxin		627D	cncl3		5C04	dead		5640
cdte	S	Macro	brkbit	S	0000	cncl4		5C19	deado		569B
cecho	S	Macro	brkcom		5833	cncluz		58CC	deadow		56B5
cend		Macro	brkmsk	S	0001	cnctry		5C26	deads		56B0
cevpar	S	Macro	bsize		0084	cnctxt		5B2A	deadw		567B
cgflow		Macro	bsytxt		5CD2	cncwon		5B7D	deadwt		5646
chand		Macro	buffw		0004	cnetxt		5CD2	deadxw		5672
chlink		Macro	bxchek		63A8	cnfirm		62C5	debbuf		0001
cinterru	S	Macro	cagrp		0000	cnfrm		62BE	dec31		6090
clink		Macro	callhl	S	6505	cnftxt		62D2	dec3r		6089
cname		Macro	carbit		0006	cnvbit		0007	decdo		6074
cnode		Macro	card0		5820	cnvmsk		0080	decin		6042
cogroup		Macro	cardrp		580B	cogrp		0000	decin1		6046
cpflow		Macro	carmsk	S	0040	cost		65FD	decin3		6064
crate		Macro	cautoh		00FF	costs		42F8	decinc		6015
crflow	S	Macro	cautol		00FF	cpad		0000	decine		602A
ctlink		Macro	caz80	S	0000	cprty		0000	decott		6076
cvlink		Macro	ccret		5CD0	crchek		62F9	decout		606B
lblk		Macro	cflow		0000	crlf		6290	decpad		6099
abort		6313	check2		5756	crltxt		59E1	deq		6913
abtabt		631E	checkw		5746	crvend		53DB	discon		641E
acc		5B27	chost		00FF	crvenq	S	53C6	dislop		59B4
acctxt		5B20	chrihr		6C98	crvlop		53B8	dism0		548E
ack		67B3	chrin		5F98	cswind	S	673B	dismax		4286
ack0		67DA	chrinb		6C95	ctab		6611	disptc		59A3
ackbit		0004	chrinh		6C7F	ctrla	S	0001	dispth		59A2
acked		6866	chrout		5FA0	ctrlc		0003	diszap		6439
ackmsk	S	0010	chrow		5FB2	ctrlg		0007	dly		6F97
ackw		684A	ciclip		0020	ctrlh		0008	dlytxt		6F3A
acntxt		6CAB	cid		5318	ctrlj		000A	dsc		5DE6
active		6B30	cintr		0000	ctrlm		000D	dscchk		636B
alntxt		6C9A	cirate		00FF	ctrlq	S	0011	dsctxt		5DDC
altiop		6CF7	clocks	S	001E	ctrls	S	0013	dsize		000D
altr		6CD3	clockv	S	0050	ctrlx		0018	dstfoh		6270
alttxt		6BFB	clp		6F86	ctype		0001	dstfou		625E
aut		6F7A	clptxt		6F28	cursp		4270	dstin		61EC
autbit		0005	clrbfi		5FC1	data0		56C5	dstin2		61F9
autmsk		0020	clrbiw		5FBE	data0a		56D1	dsto2		624F
auttxt		6F1C	clrend		5388	data1		56DA	dstout		6247
b110	S	0002	clrlop		5378	data2		5700	dtndce		57C0
b1200	S	0007	cm0tab		59EE	data2a		570A	echbit		0005
b134p5	S	0003	cmd		5918	datac2		5736	echin		5FF6
b150	S	0004	cmdbit		0006	datac9		573C	echinl		6002
b1800	S	0008	cmdbkl		593E	datacd		572A	echl	S	5F43
b19200	S	000F	cmdbkp		5932	datam2		5719	echmsk	S	0020
b2000	S	0009	cmdchk		5F78	datam9		571E	echr		5F4E
b2400	S	000A	cmdmsk		0040	dataml		5715	eclbit		0007
b300		0005	cmdtim		00B4	datar		57A2	eclmsk	S	0080
b3600	S	000B	cmetxt		5D22	dattim		0004	eco		6FA6

ecotxt		5E4E	gpiadd		61A2	itb		70AF	lsize		0080
empty		54D9	gpidon		61A9	itbtxt		7083	lsp		000F
enq		695C	gpihlp		61B1	itc		708C	lstack		007F
error		595F	gpilop		61BB	itctxt		7071	ltdown		0012
errorp		595E	gpizro		61A5	itn		70AB	ltime		0011
errorq	S	5965	gpolop		61D9	itntxt		7088	ltype1		0000
etdtxt		5CE6	gpoout		61E0	itran		554F	ltype2		0001
exchad		68A2	gretxt		5D4A	ivtn		531C	lwatch		001A
ff		0001	grp		7002	ixoffs	S	002F	major		0000
ffs		43CB	grpin		6187	iyend		64D4	map		5A24
flb		6FF7	grpout	S	61D2	iylin		64DE	mapl2		5A81
flbtxt		6FE5	grptxt		6F48	iynext		64D6	mapl3		5A88
flg		6FF0	gtime		5533	iynum		698B	maplnk		5A50
flgtxt		6FDD	h16out	S	6C3B	jmptxt		6C00	maplop		5A2F
fln		6FFF	h8out		6C40	jump		6D62	maplv		5A92
flntxt		6FE9	han		7011	kil		5E05	mapnlp		5A40
flo		6FB7	hantxt		6F4F	kill		66A7	mapnod		5A3E
flotxt		5E37	hbit		0002	kiltxt		5E01	mapout		5A6D
flp		6FFB	hedbit		0005	lagrp		0008	maptxt		5A21
flptxt		6FD7	hedmsk	S	0020	lautoh		0005	marger	S	00C7
frleng		427D	hex2		6C5A	lautol		0006	match		59C8
frlist		4279	hexc		6C4F	lblk0		431B	mcerr		00C0
g0bit	S	0000	hexin		6C5D	lblk1	S	439B	mcli		0009
g0msk	S	0001	hexinf		6C6B	lblk2	S	441B	mclo	S	000A
g1bit	S	0001	hexinl		6C66	lblk3	S	449B	mcmder	S	00C0
g1msk	S	0002	hmsk		0004	lblk4	S	451B	mdial		001A
g2bit	S	0002	hnc		704C	lcid		0017	mdieer	S	00C6
g2msk	S	0004	hnctxt		7031	lclip		0003	me		4200
g3bit	S	0003	hnd		704F	ldbeg		0000	mem0lp		6535
g3msk	S	0008	hndtxt		703B	ldbtim		0016	mem1lp		651F
g4bit	S	0004	hnetxt		5D05	ldend		000C	memch0		651E
g4msk	S	0010	hns		7053	leds		6981	memch1		654A
g5bit	S	0005	hnstxt		7042	lflag1		0014	memchk		6507
g5msk	S	0020	hnt		7048	lflag2		0015	memcok		6519
g6bit	S	0006	hnttxt		7036	lhost		0004	mer		0005
g6msk	S	0040	hold		58B6	lights		00FE	merr		0080
g7bit		0007	icarve	S	53A3	lik		7082	mew		0006
g7msk		0080	iclear	S	5378	liktxt		6F58	mftbit		0006
gbit		0003	idled		5FD8	lin		001C	minor		0004
gcid		6979	idltxt	S	5FDB	lintr		000C	mkl		0007
gcid1		697C	iend	S	542F	lkitxt		6224	mliner	S	00C3
geqmin		6597	ihedr		543D	ll		001F	mmtyer		0084
getbf0		68C4	ijob	S	5404	llyank		002D	mnrmer		0082
getbff		68E8	ijobl		540A	lmymsk		000E	mnull	S	0000
getbf1		68CF	ijobml	S	5428	lnchek		638A	move		6685
getblk		722D	imemc2		5391	lnkin		620A	mpc0		00F0
getbuf		68DA	imemcl		538D	lnum		000D	mpc7	S	00F7
getlok		63C4	imgin		5F92	logrp		0007	mpc99		00FF
getreg		686C	imginw		5F80	lout		001B	mpcbka		6AB9
gfxbit		0007	impc	S	5343	low7		007F	mpcdo		69E8
gfxmsk		0080	implop		5348	lpad		000B	mpcdos	S	6A02
givbuf		6902	impnxt		536F	lprty		0009	mpcic		6A87
givlok		6412	impsll		5361	lrate		000A	mpcice		6AA2
gmsk	S	0008	int		7056	lrcbit		0005	mpcin		6A0C
gonbit		0007	inttxt		5E44	lrcmsk	S	0020	mpcin2		6A1F
gonmsk	S	0080	ioptxt		6CA1	lremb		0018	mpcin3		6A37
gontst		5878	iopw		0008	lreml		0019	mpckbc		6AEE
gotlok		6408	irut	S	53E7	lrut		001D	mpcnmi		6B15

mpcnsh		6B55	mwprty		0012	pktded		55E1	ratin		60A6
mpcnum	S	698E	mwpx		0014	pkther		0020	ratout		6106
mpcnv		6B5D	mwt		0003	pktin		5576	rcall		6802
mpcoc		6AA6	mymsk		0001	pktini		556C	rcall0		681D
mpcou2		6A5D	name		4206	pktins		5573	rdata	S	4018
mpcou3		6A76	name0		4203	pktlop		5616	rdie		F003
mpcout		6A4A	nbox		0010	pktme		0010	rdstxt		646C
mpcptl		6BC3	ncttxt	S	58EA	pktnxl		5626	record	S	4014
mpcrat		6ADE	netgon		5880	pktrup	S	55A1	rel		6688
mpcsel		6995	netrbl		58C9	pktsup		55E7	rellop		66BC
mpcsh		6B4D	newrut		4287	pkttim		55CE	relnxt		6705
mpcsla		6998	next		596E	plen		0005	remchk		58E3
mpcsta		6ACE	next2		5997	plink		0004	remerr		58E7
mpcuns		69A3	nextp		6D46	plroff		0014	remtxt		5DA4
mpcwat		6986	nextq		5975	pmax		007F	remup		5905
mpcwte		69D8	nlink		0020	pminus		0009	rerret		685B
mpcwti		69CE	nme		000F	pmttxt		59E9	rets		63A6
mpcwtl		69C8	no		62F4	poret		6857	richlt	S	4181
mpcwtr		69E4	nodeup		5908	ppl		6621	richa	S	405C
mpczpl		6B3C	noncnf		62FC	pplcnv	S	671C	richb	S	405E
mpunv		0004	notxt		62EF	ppldon	S	6642	richb2	S	40DE
mr3		000C	novl		6BC0	pplhim	S	6660	richc	S	405B
mrbaud		000D	ntab		58D3	pplist		4275	richs	S	4180
mrcbk		0019	nulerr		6C8E	ppllit		54DB	richt	S	417F
mrclip	S	0017	nultxt		59E3	pplme		670B	riolin	S	4059
mrecho	S	000F	nxtj0		5486	pplmor		664B	riompc	S	4058
mrgx	S	0015	nxtjob		5478	pplmty		0002	rmpc	S	405A
mrint	S	001D	nxtlop		54A6	pplout		667B	rngbit		0006
mrn		0008	nxtret		5481	pplrup		66A1	rngmsk	S	0040
mrp		0002	nxtsl0		5490	prclen		000F	rngtim		000A
mrpad	S	001B	ofetxt		5D86	prega		000C	routes		42EB
mrprty	S	0011	oldclk		4274	pregb		000E	rram		4000
mrpx	S	0013	onbit		0003	pregc		000F	rsize	S	4014
mrster		0085	onmsk		0008	pregd		0010	rsm		5DF8
mrt		0001	paddrh		000B	prege		0011	rsmtxt		5DF2
msave	S	2200	paddrl		000A	pregf		000D	rstack		417D
msbk		0008	pae		70DD	pregh		0012	rtierr	S	60C2
mtyper	S	00C1	paetxt		70D2	pregl		0013	rtihlp		60D7
munv		0000	pan		70E3	prllen		0004	rtihnt		60CB
mv1		2002	pantxt		70D9	prup0		0007	rtilop		60BC
mv2		2003	pao		70E0	prup0x		55C6	rtimat		60C5
mv3		2004	paotxt		70D6	prupb		0017	rtimem	S	417E
mvc		2000	par		70BB	prupl		0018	rtitab		60F6
mvend	S	200D	partxt		6F60	pst		6ECD	rtohnt		6121
mvl	S	2001	patch	S	7248	pstlen		001C	rtoout		611B
mvtime		200D	patrn		4182	psttab		6EDA	rtotab		6127
mvx1a		2005	pbakh		0003	psttxt		6C30	rtype	S	4017
mvx1c		2007	pbakl		0002	ptob		0006	rubout		007F
mvx2a		2009	pbkbit		0006	ptol		0007	runit		551D
mvx2c		200B	pbkmsk		0040	ptx1ch		0009	runpaf	S	5531
mwbaud		000E	pcid		0008	ptxnch		007B	runv	S	F006
mwclip		0018	pforh		0001	ptxovl		0003	rup		654B
mwecho		0010	pforl		0000	pvchek		632B	rupgtr		659B
mwgx		0016	pfrb		0008	quest		5DB5	ruplop		6563
mwind		2000	pfrl		0009	questl		5DB6	ruplp2		6575
mwint		001E	pfxbit		0006	questq		5DBA	ruplpi		6573
mwp		0004	pfxmsk		0040	quests		5DD5	ruplss	S	657C
mwpad		001C	pktaru		55C0	raddr	S	4015	rupnum		0002

```
rupnxt      65E8    strout      6293    vupmsk      0008    xmapy     S 5A8C
rupscn    S 65AA    strowt      62B9    wait        5785    xnsg        5E97
ruptim      0002    subr        6785    waitdb      5790    xsetdo      7196
sbrret      67A2    sync        5885    waitdc    S 5796    xsetit      7242
scngtr    S 65C9    syssp       426E    waitdl      5791    xsfb      S 5EC1
scnlop      6583    tbltxt      5883    waitml      578D    xsfg      S 5EA2
scnnxt      65D2    tdownp      64F7    watch       6DB8    xsfn      S 5EC8
scnpop      65E3    temp1       4272    watdsc      6E02    xsfp      S 5E95
select      00FF    text        675D    watdwn      6E10    xsg         5EAA
sendit    S 5757    therr       7165    watend      6E84    xsib      S 5F0A
setblk      722D    thonly      718A    watlop      6DD1    xsic        5F32
setnz       6BC6    timt2       57F9    watnlp      6DE5    xsiv      S 5F3B
setnz0      6BC9    timtgo      57DC    watnod      6DE6    xsync       58A7
setreg      6887    todata      5BA3    watret      6E2F    xwatch      6E9C
settd       64FE    todead      5FF0    watrut      6E44    xwatlp      6EA2
sfb         5EBE    todedo    S 648A    watstr      6E13    xwatnx      6EC5
sfbtxt      5E87    todedp      6488    watth     S 6DF9    xysg        5EA3
sfg         5E9F    totbuf      427F    wattop      6DC7    yankhm      64A1
sfgtxt      5E7F    totxt       59E4    wattxt      6C1C    yankme      648D
sfltxt      5E37    tryluz      5C3F    watv        6E07    yes         62F1
sfn         5EC5    tyh         7151    wcotxt      6D92    yesno       62DE
sfntxt      5E8B    tyhtxt      7131    wdctxt      6D99    yestxt      62EC
sfp         5E92    tyn         715F    wdntxt      6DAC    z80as     S 0001
sfptxt      5E78    tyntxt      7142    whchek      62F6    zapech      5466
shbit       0001    typ         70EF    whetxt      6D70    zaplin      5449
shmsk       0002    typtxt      6F6D    whrtxt      6CA7
si2txt      5EF5    tyt         7155    wittxt      6CB7
sib         5F07    tyttxt      711C    wrtlop      6E4D
sibtxt      5EDE    tyv         7245    wrtnxt      6E76
sic         5F19    tyvtxt      7113    wrttxt      6E31
sictxt      5EE3    unschk      69AE    wuptxt      6DA0
sindo       5F0F    upc         5ADF    wwbit       0002
sintxt      5E44    upc0        5B0A    wwmsk     S 0004
sit         5E10    upd         7162    xbreak      586C
sitecc      5F3E    updtxt      6F72    xcl         5F72
sitflo      5E60    vbit        0004    xclhi       6BDD
sitint      5ECB    vbuf0       7288    xcllok      4288
sitlop      5E18    vcheck      4283    xcnadr      5C5F
sitrem      5E53    vdie        4003    xcnerr      5CD0
sitrst      5F5B    vdieaf    S 4008    xcnho       5C44
sittxt      5E0C    vdiebc    S 400A    xcnlop      5C48
siv         5F38    vdiede    S 400C    xcnnds      5C8C
sivtxt      5FC2    vdiehl    S 400E    xcnnxt      5C59
skip        6632    vdieix    S 4010    xcnwon      5CA6
skip1       6638    vdieiy    S 4012    xdeps       6D56
slolop      5501    vdiepc    S 4004    xdouns      5EAC
slonxt      5515    vdiesp    S 4006    xdsc        6441
slow        4285    vers        4201    xech        5F56
slowsb      54FB    vertxt      59D5    xechl     S 5F46
slowtc      003C    vlast       4281    xechr     S 5F51
snntxt      71F4    vletxt      5D68    xexmn       6D4B
spd         70E6    vmsk        0010    xhold       58CE
spdtxt      6F67    vpat        0040    xmap        5A9F
srmtxt      5E30    vram        4200    xmap2       5AC4
stack       42C9    vrom      S FCC0    xmape       5AD5
stroth      628A    vtn0      S 4000    xmapf       5ADD
stroul      62A2    vtnup       0080    xmapn       5A9E
strous      62A8    vupbit      0003    xmapt       5AB7
```

```
Assembly began:       Thu Sep 23 15:27:12 1982
Assembly finished:    Thu Sep 23 15:51:17 1982
Assembly real time:   1445 seconds.
Processor time used:  user 346.32, system 8.03  (total 354.35, 25% of cpu.)
Defined 901 symbols.
Used 443 of 511 hash buckets.
Processed 5493 real source lines for 228 lines/minute.
Macros expanded into 11003 lines.
Effective source size 16496 lines for 685 lines/minute.
Defined 28 macros; stored 108 macro lines.
Processed 2013 macro calls; expanded 2920 parameters.
```