Everything You Wanted to Know About Interfacing to the HP3000 PART I

Ross Scroggs
The Type Ahead Engine Company
Oakland, California

INTRODUCTION

It is important to realize that the information presented in this paper is my interpretation of the facts. The interpretation is not perfect, for surely I have included incorrect statements. If you believe that something here is incorrect, bring it to my attention. If I believe that you are wrong I will try to set you straight, but I will not argue about anything. I have included a list of references at the end of this paper from which I have obtained most of the information included here. If you desire to make all of your terminal attachments successful, obtain all of the references and read them. The most important piece of information I can give you is to start planning early when attaching terminals to the HP3000 and don't believe anything you read, if you haven't seen it work yourself, plan on having to solve a few problems. This paper is a guide to solving those problems, but it won't solve them for you.

Most of the experiments outlined in this paper were performed with the Bruno release of MPE-IV, I have subsequently been informed that the C release of MPE-IV fixed many terminal driver problems associated with the ADCC.

Asynchronous terminals are attached to the HP3000 Series I, II, and III through the Asynchronous Terminal Controller (ATC) and to the Series 30, 33, 40, and 44 through the Asynchronous Data Communications Controller (ADCC). This paper addresses issues involved in making a successful connection to one of these two devices. Terminals attach to the Series 64 through the Advanced Terminal Processor (ATP) which should make all of our lives simpler (though expensive) in the coming years. In its earlier versions the ATP will act much like the ATC in terms of interfacing to terminals. It features two major advances over the previous terminal controllers. First, there is a microprocessor controlling each terminal line, this removes considerable work from the CPU, the "character interrupt" problem. Second, the ATP can use either the RS-232 or RS-422 interface standards. RS-422 is a completely new electrical and mechanical interface that supports very high data rates over great distances with no errors, a typical example would be 9600 baud at 4000 feet. What this flexibility costs you is about $200 extra per terminal to provide a RS-232 to RS-422 adapter. These won't be required when terminals provide RS-422 interfaces.

Terminals attached to the ATC or ADCC are accessed primarily in two ways: as a session device or as a programmatically controlled device. A session device is one on which a user logs on with the HELLO or () commands and accesses the HP3000 through MPE commands. A programmatic device is one which is controlled by an application program that is run independently from the device. These two access methods are not mutually exclusive, a session device can be accessed programmatically and many MPE commands can be executed on behalf of a user who is accessing the system programmatically.

SESSION DEVICES

Attaching a terminal as a session device is typically the easier of the two methods. You must set the terminal speed, parity, subtype, and termtype correctly and provide the proper cable to complete the hookup.

Terminal Speed

The speeds supported by the ATC are 110, 150, 300, 600, 1200, and 2400 baud. The speeds supported by the ADCC are those of the ATC plus 4800 and 9600 baud. Unfortunately these two higher speeds can not be sensed by the ADCC and thus you must log on at a lower speed and use the MPE SPEED command to access the higher speed. (Use of subtype 4 and specifying any speed will allow a terminal to log on at that speed only, this includes 4800 and 9600. Note however, that if you use the :SPEED command the new speed specified will be required at your next logon.)

Terminal Parity

The format of characters processed by the HP3000 is a single start bit, seven data bits, a parity bit, and one stop bit (two at 110 baud). The parity bit may always be zero, always be one, computed for odd parity, or computed for even parity. Choosing the proper parity setting has been complicated by differences between the ATC and ADCC. The ATC inspects the parity bit of the initial carriage return received from the terminal and sets parity based on that bit. If the bit is a zero the ATC generates odd parity on output, if it is a one the ATC generates even parity on output. In either case the parity of incoming data is ignored and the parity bit is always set to zero before the data is passed to the requesting program. The ADCC also sets parity based on the parity bit of the initial carriage return but does so with a slight, but nasty twist. If the bit is a zero the ADCC passes through the parity bit supplied by the application program on output, if it is a one the ADCC generates even parity on output. If pass through parity was selected the parity of the incoming data is passed through to your program buffer. If even parity was selected the input data is checked for proper even parity. Thus, you should not use odd or force to one parity on the ADCC. The odd parity will be interpreted as pass through and the parity bits will wind up in your data buffer, string comparisons will fail because of the parity bits. Force to one parity will be interpreted as even and all input will cause parity errors.

Subtype

The ATC supports subtypes 0, 1, 2, 3, 4, 5, 6, 7, the ADCC support subtypes 0, 1, 2, 3, 4, 5. Subtypes 2, 3, 6, 7 concern half duplex modems and not me, so I will ignore them. Subtype 0 is the standard for directly attaching terminals without modems. (Note that terminals that are attached to multiplexors can fit in this category, the modem involved is managed by the multiplexor, not the HP3000.) Subtype 1 is the standard for attaching terminals that use full duplex modems such as Bell 103, 212 and Vadic 34xx. Both subtypes 0 and 1 speed sense on the initial carriage return. Subtype 4 is for direct attach terminals that will not be speed sensed, they will run at a fixed speed that is set at configuration time. This subtype is often used to prevent the HP3000 from trying to speed sense garbage, this sometimes occurs when using short-haul modems (line-drivers) that do not have a terminal attached to the other end. Subtype 5 is for modem attached terminals that will not be speed sensed.

Termtype

The ATC supports terminal types 0, 1, 2, 3, 4, 5, 6, 9, 10, 12, 13, 15, 16, 18, 19, 31, the ADCC supports terminal types 4, 6, 9, 10, 12, 13, 15, 16, 18, 19. Termtype 4 is for Datapoint 3300 terminals, it outputs a DC3 at the end of each output line and responds to backspace with a Control-Y, truly bizarre. (Termtype 4 on the ADCC does not output DC3s at the end of each line.) Termtype 6 is for low speed printers, it outputs a DC3 at the end of each line but responds to a backspace with a linefeed. (The linefeed is on the first backspace of a series, this allows you to type corrections under the incorrect characters.) Termtype 9 is the general purpose non-HP CRT terminal type. No DC3s are output at the end of the line (whew!!) and nothing strange happens on backspace, the cursor backs up just as you would expect. Termtype 10 is the standard for HP-26xx terminals. Termtype 13 is typically for those terminals at a great distance from the HP3000 for which some local intelligence echos characters and the 3000 should not. (Telenet and Tymnet charge you for those echoed characters, that's reason enough not to have the HP3000 echo them.) Termtypes 15 and 16 are for HP-263x printers. Termtype 18 is just like termtype 13 except that no DC1 is issued on a terminal read. Certain termtypes less than 10 specify a delay after carriage control characters are output to the terminal. The ATC handles this by delaying for the designated number of character times but does not output any characters. The ADCC actually outputs null characters. The most extreme case is termtype 6 which causes 45 nulls to be output after a cr/lf at 240 cps.

Cable

Direct attach terminals, subtypes 0 and 4, use only three signals in the cable: pin 2, Transmit Data, pin 3, Receive Data, and pin 7, Signal Ground. (Note that all signal names are given from the point of view of the terminal, not the modem or the HP3000 which acts like a modem.) Typically the cable will connect pin 2 at the terminal end to pin 2 at the HP3000, pin 3 at the terminal to pin 3 at the HP3000 and pin 7 at the terminal to pin 7 at the HP3000. This is not to say that your terminal does not require other signals, it just says that the HP3000 is not going to provide them, you must. If your terminal requires signals like Data Set Ready, Data Carrier Detect, or Clear To Send, you can usually supply these signals to the terminal with a simple cable patch. Jumper pin 4, Request To Send to pin 5, Clear To Send. Jumper pin 20, Data Terminal Ready to pin 6, Data Set Ready and pin 8, Data Carrier Detect. These two jumpers cause the terminal to supply its required signals to itself.

Modem attach terminals, subtypes 1 and 5, use seven signals in the cable: pin 2, Transmit Data; pin 3, Receive Data; pin 4, Request To Send; pin 6, Data Set Ready; pin 7, Signal Ground; pin 8, Data Carrier Detect; and pin 20, Data Terminal Ready. Naming the signals gets complicated since the HP3000 is acting like a modem and it is being attached to a modem. Typically, the cable that connects the HP3000 to the modem will connect pin 2 at the modem end to pin 3 at the HP3000, pin 3 at the modem to pin 2 at the HP3000, pin 4 at the modem to pin 8 at the HP3000, pin 6 at the modem to pin 20 at the HP3000, pin 7 at the modem to pin 7 at the HP3000, pin 8 at at the modem to pin 4 at the HP3000, and pin 20 at the modem to pin 6 at the HP3000.

The cable that attaches your terminal to a modem should be specified in your terminal owners manual, consult it for proper connections.

Flow Control

Flow control is the mechanism by which the speed/amount of data from the HP3000 to the terminal is controlled. The HP3000 supports two flow control methods, ENQ/ACK and XON/XOFF. The ENQ/ACK protocol is controlled by the system, after every 80 output characters the systems sends an ENQ to the terminal and suspends further output until and ACK is received back from the terminal. The suspension is of limited duration for termtypes 10 to 12, output resumes if no ACK is received in a short amount of time. The suspension is indefinite for termtypes 15 and 16, the ENQ is repeated every few seconds until an ACK is received. (It is the ENQ/ACK protocol that fouls up non-HP terminals that attempt to access the HP3000 through a port that is configured for an HP terminal. Most terminals do not respond to an ENQ with an ACK, you must do it manually by typing Control-F which is an ACK. An ENQ is generated by the HP3000 when the initial carriage return is received from the terminal, thus you get hung immediately. But, hit Control-F, and logon and specify the proper termtype in your HELLO command.)

The XON/XOFF flow control protocol is controlled by the terminal. When the terminal wishes to suspend output from the HP3000 it sends an XOFF (Control-S or DC3) to the HP3000 and sends an XON (Control-Q or DC1) to resume output. Unfortunately the HP3000 sometimes fails to properly handle one of the two characters and you either overflow your terminal or get hung up. This is particularly nasty when your terminal is a receive-only printer and you can't supply a missing XON. You're really dead if the HP3000 misses the XOFF. Termtype 13 has in my experience been the best termtype to use if your terminal requires the XON/XOFF flow control protocol. You can turn the echo back on with ESC :.

A special note on XON. If you inadvertently send an XON (DC1) to the HP3000 when output is not suspended, surprise you are now in paper tape mode and backspace, Control-X, and linefeed will act most strangely. Hit a single Control-Y to get out of this mode, the Control-Y will not be received by your program.

Some terminals perform flow control by raising and lowering a signal on their interface, the HP3000 can not handle this. You must either run the terminal at a low enough speed to avoid overflowing it or provide hardware to convert the high/low signal to ENQ/ACK or XON/XOFF, a costly affair.

A form of flow control used by HP terminals when inputting data to the HP3000 is the DC2/DC1 protocol. When the enter key is pressed on the terminal, a DC2 is sent to the HP3000 to alert it to a pending block mode transfer. When the HP3000 is ready to receive the data it sends a DC1 back to the terminal to start the data transfer. (Your program does not handle the DC2/DC1, but see below FCONTROL 28, 29.) This works fine except in certain circumstances. In certain modes the HP actually sends DC2 carriage return when the enter key is pressed. This is no problem unless the DC2 and CR do not arrive together. The CR may be seen as the end of the data if it comes sufficiently far behind the DC2, your program completes its request for data with nothing and the real data bites the dust when it finally shows up. The separation of the DC2 and CR can occur when using statistical multiplexors or when using Telenet or Tymnet. Be aware, this problem is infrequent, but unsettling when it occurs.

PROGRAMMATIC DEVICES

Attaching a terminal as a programmatic device is usually done when you want to attach a serial printer, instrument, data collection device, or other strange beast to the HP3000. An application program you write will typically control all access to the device, a user will not walk up to it, hit return, and log on. I will explain the various intrinsics that are used to access programmatic devices and will give short (incomplete) program segments that illustrate the access method.

Declarations

The following declarations will be assumed for all program segments shown.

```
begin

integer
    ilen,
    olen,
    pfnum:=0,
    pifnum:=0,
    pofnum:=0,
    precsize:=-256;  <<** pick a number large enough for the
                          maximum data transfer **>>

logical
    fcontrol'parm:=0,
    prev'echo;

logical array
    ibuff'(0:255),
    obuff'(0:255),
byte array
    pfname(0:7):="PROGDEV ";

byte array
    pdevice(0:7):="PROGDEV ";

byte array
    pifname(0:7):="IPROGDV ";

byte array
    pidevice(0:7):="IPROGDV ";

byte array
    pofname(0:7):="OPROGDV ";

byte array
    podevice(0:7):="OPROGDV ";

define
    fs'error'on'ccl= if <  then file'error(#,
    fs'error'on'ccne=if <> then file'error(#;

procedure print'message(enum);
  value
    enum;
  integer
    enum;
  option external;

intrinsic
    fclose,fcontrol,fopen,fread,fsetmode,fwrite,print'file'info,
    getprivmode,getusermode,iowait,terminate;

subroutine file'error(fnum,enum);
  value
    fnum,enum;
  integer
    fnum,enum;
  begin

    <<** simple file error handling subroutine, basic, not fancy
         or very good. **>>

    print'file'info(fnum);
    print'message(enum);   <<** supply something, but remember your
                                cliches, make it user-friendly! **>>
    terminate  <<** simple, direct, not too graceful **>>
  end;  <<* file'error *>>
```

FOPEN

You must call FOPEN to gain access to the device, I always use a formal file name to allow control of the open with file equations. If the device is unique in the system, I use its device name as the file name. The foptions specify CCTL, undefined length records, ASCII, and a new file. The aoptions specify exclusive access and input/output. Choose a record size that is larger than the maximum data transfer that will take place.

ATC — Opening a terminal with an HP termtype causes an initial ENQ to be output to the device on the first output, there must be an ACK reply from the device or your program will wait until the ENQ time-out occurs.

ADCC

For devices that are to be used exclusively in programmatic mode it is recommended that you REFUSE the device so that extraneous carriage returns from the device will not be speed sensed by the HP3000.

```
pfnum:=fopen(pfname,%604,%104,precsize,pdevice);
fs'error'on'ccl(pfnum,1);
```

FCLOSE

You call FCLOSE to release access to the device, some FCONTROL options exercised while the device was open are not reset by FCLOSE.

ATC — MPE sends a cr/lf to the device if it believes that the "carriage" is not at the beginning of the line, i.e., the last character output was not a linefeed.

ADCC — MPE sends a cr/lf to the device if it believes that the "carriage" is not at the beginning of the line, i.e., the last character output was not a linefeed or formfeed.

```
fclose(pfnum,0,0);
fs'error'on'ccl(pfnum,9);
pfnum:=0;  <<** I do this for error handling purposes **>>
```

FREAD

You call FREAD to get data from the device, many of the FCONTROL calls shown below affect how FREAD works. End-of-file is indicated by a record that contains ":EOF:". Any record with a colon in column one is an end-of-file to $STDIN, ":EOD", ":EOJ", ":JOB", ":DATA", and ":EOF:" are end-of-file to $STDINX. You should avoid linefeeds that follow carriage returns because garbage characters will be echoed to the terminal. (The inbound linefeed collides with the outbound linefeed coming as a result of the carriage return.)

```
ilen:=fread(pfnum,ibuff',precsize);
fs'error'on'ccl(pfnum,2);
if >
  then ; <<** handle eof **>>
```

You may want to trap certain errors returned by FREAD to your program: 22, software time-out; 31, end of line (alternate terminator); and 33, data lost.

ATC — The characters NULL, BS, LF, CR, DC1, DC3, CAN (Control-X), EM (Control-Y), ESC, and DEL are stripped from the input stream for both session and programmatic devices.

ADCC — The characters BS, LF, CR, CAN (Control-X), and EM (Control-Y) are stripped from the input stream for session devices. The characters BS, CR, and CAN (Control-X) are stripped from the input stream for programmatic devices.

The default parity cases are handled quite differently between the ATC and ADCC, you should exercise extreme caution when dealing with parity on the ADCC.

ATC — If the ATC is in the odd/out, no check/in mode all incoming characters have their parity bits set to zero. The same is true for even/out, no check/in mode.

ADCC — If the ADCC is in pass thru/out/in mode all incoming characters retain their parity bits, they are not set to zero. All special characters must have a zero parity bit to be recognized. If the ADCC is in even/out, check even/in mode the incoming characters must have proper even parity and their parity bits are set to zero. The second time you open this terminal the ADCC has switched to pass thru mode and all incoming characters retain their parity bits!!!

Each time you issue an FREAD to the terminal MPE sends a DC1 to the terminal to indicate that it is ready to accept data. Most devices ignore, totally, the DC1. If your a device reacts negatively to the DC1, use termtype 18 which suppresses the DC1 on terminal reads. The device must not send data to the HP3000 until it has received the DC1, otherwise the data will be lost. If the device does not wait for the DC1 you must supply external hardware that will provide buffering and wait for the DC1 or you can solve the problem on the HP3000 by using two ports to access the device. One port is opened for reading and the other for writing. A no-wait read is issued before the write that causes the device tc send data, then the read is completed.

```
getprivmode;  <<** necessary for nobuf, no-wait i/o **>>
pifnum:=fopen(pifname,%204,%4404,precsize,pidevice);
if <
  then begin
    getusermode;
    file'error(pifnum,1)
  end;
getusermode;
pofnum:=fopen(pofname,%604,%404,precsize,podevice);
fs'error'on'ccl(pofnum,1);
...
ilen:=fread(pifnum,ibuff',precsize);
fs'error'on'ccl(pifnum,2);
fwrite(pofnum,obuff',-olen,%cctl);
fs'error'on'ccne(pofnum,3);
iowait(pifnum,ibuff',ilen);
fs'error'on'ccne(pifnum,22);
```

When you attach your device to the two ports, connect pin 2, Transmit Data of the terminal to pin 2 of the read port, connect pin 3, Receive Data of the terminal to pin 3 of the write port, and pin 7, Signal Ground of the terminal to pin 7 of both ports. (This two port scheme was first introduced to me by Jack Armstrong and Martin Gorfinkel of LARC.)

FWRITE

You call FWRITE to send data to the device. The carriage control (cctl) value of %320 is often used to designate that MPE send no carriage control bytes, such as cr/lf, to the device. Some FCONTROL calls shown below affect how FWRITE works. Control returns to your program from FWRITE as soon as the data is loaded into the terminal buffers, it does not wait until all data has been output to the device.

ATC — Carriage control %61 is output as carriage return, formfeed (termtype 10).

ADCC — Carriage control %61 is output as formfeed (termtype 10).

The default parity cases are handled quite differently between the ATC and ADCC, you should exercise extreme caution when dealing with parity on the ADCC.

ATC — If the ATC is in odd/out mode all outgoing characters are given odd parity, even parity is generated when the mode is even/out. Simple.

ADCC — If the ADCC is in pass thru/out mode all outgoing characters retain their parity bits as passed to FWRITE. If the ADCC is in even/out mode all outgoing characters are given even parity. The second time you open this terminal the ADCC has switched to pass thru/out and all outgoing characters retain their parity bits!!!

```
    fwrite(pfnum,obuff',-olen,%cctl);
    fs'error'on'ccne(pfnum,3);
    <<** eof here is probably an error, I mean what is going on? **>>
```

FSETMODE — 4 — Suppress carriage return/linefeed

In normal operation a line feed is sent to the terminal if the input line terminates with a carriage return, a cr/lf is sent to the terminal if the line terminates by count, and nothing is sent if the line terminates with an alternate terminator. FSETMODE 4 suppresses these linefeeds and carriage returns. FSETMODE 0 returns to normal line termination handling, an FCLOSE also returns the device to the normal mode.

```
      fsetmode(pfnum,4);
      fs'error'on'ccl(pfnum,14);
```

FCONTROL

FCONTROL is the workhorse intrinsic for managing a programmatic device on the HP3000. Each use of FCONTROL which be shown separately but it will usually be the case that several calls will be used. Most calls are required only once, but the timer calls are required for each input operation. Each call will be identified by the controlcode parameter that is passed to FCONTROL.

FCONTROL – 4 – Set input time-out

This option sets a time limit on the next read from the terminal. It should always be used with devices that operate without an attached user to prevent a "hang." If something goes wrong with the device, your program will not wait forever, control will be returned to your program. The FREAD will fail and a call to FCHECK will return the errorcode 22, software time-out. No data is returned to your buffer in the case of a time-out, any data entered before the time-out is lost. If you issue a timeout for a block mode read the timer is stopped when the DC2 is received from the terminal, a new timer is then started which is independent of the timer set by this FCONTROL call. See the section below on enabling/disabling user block mode transfers.

```
    fcontrol'parm:=30;  <<** 30 second time-out **>>
    fcontrol(pfnum,4,fcontrol'parm);
    fs'error'on'ccl(pfnum,413);
    ilen:=fread(pfnum,ibuff',precsize);
    if <
      then begin
        fcheck(pfnum,errorcode);
        if errorcode <> 22
          then file'error(pfnum,errorcode*100+2);  <<** something else **>>
        <<** handle time-out **>>
      end;
```

FCONTROL – 10, 11 – Set terminal input/output speed

These FCONTROL options allow you to change the terminal input and output speeds. FCONTROL 37 can also be used to set terminal speed, it sets termtype as well and is the method that I prefer.

ATC — Split speeds are allowed.

ADCC — Split speeds are not allowed, FCONTROL 10 and 11 set both input and output speed.

FCONTROL – 12, 13 – Enable/disable input echo
These FCONTROL options allow you to enable and disable terminal input echoing. Many devices that attach to the HP3000 do not expect or desire echoing of the characters they transmit. This option along with FSETMODE 4 completely turns off input echoing. (Control-X is handled separately.) Echoing is not restored when a file is closed so you should always put echo back the way it was found.

```
        fcontrol(pfnum,13,prev'echo);
        fs'error'on'ccl(pfnum,1313);
        ...
        <<** turn echo back on if it was previously on **>>
        if prev'echo = 0
          then begin
            fcontrol(pfnum,12,prev'echo);
            fs'error'on'ccl(pfnum,1213)
          end;
```

FCONTROL – 14, 15 – Disable/enable system break
The break key is typically disabled when terrible things will happen if the user hits break and aborts out of a program. You, the programmer, always seem to need break for debugging purposes and discover that you have it turned off. System break can only be enabled for session devices, it is not allowed for programmatic devices. If break is entered on a session device the data already input will be retained and provided to the user program after a resume and the completion of the read. If a break is entered on a programmatic device a null will be echoed to the device but no data is lost.

FCONTROL – 16, 17 – Disable/enable subsystem break
Subsystem break is recognized only on session devices, it can be enabled on programmatic devices but has no effect. If a Control-Y is entered during a read, the read terminates and the data already input will be retained and provided to the user program after the Control-Y trap prodedure returns. If Control-Y is disabled any Control-Y will be stripped from the input but no trap procedure is called and the read continues. Control-Y trap procedures are armed by the XCONTRAP intrinsic. A subsystem break character other than Control-Y may be specified when unedited terminal mode (FCONTROL 41) is used.

ATC — In programmatic mode Control-Y's are always stripped from the input.

ADCC — In programmatic mode Control-Y is not stripped from the input if subsystem break is enabled.

FCONTROL – 18, 19 – Disable/enable tape mode
ATC — This is effectively an FSETMODE 4, an FCONTROL 35, and suppression of backspace echoing all rolled into one.

ADCC — Tape mode can not be enabled.

FCONTROL — 20, 21, 22 — Disable/enable terminal input timer, read timer
These options can be used to determine the length of time it took to satisfy a terminal read. It is not a timeout, that is FCONTROL 4. The manual states that you must enable the timer before each read so why is there a disable option? If you read the timer without enabling the timer, you get the time of the most recent read that did have the timer enabled. The number returned is the length of the read in one-hundreths of a second. Condition code > implies that the read exceeded 655.35 seconds.

```
fcontrol(pfnum,21,fcontrol'parm);
fs'error'on'ccl(pfnum,2113);
ilen:=fread(pfnum,ibuff',precsize);
fs'error'on'ccne(pfnum,2);
fcontrol(pfnum,22,fcontrol'parm);
fs'error'on'ccl(pfnum,2213);
```

FCONTROL – 23, 24 – Disable/enable parity checking
This option enables parity checking on input for the parity sense specified by FCONTROL 36. Parity checking is overridden by binary transfers (FCONTROL 27) or unedited mode (FCONTROL 41).

ATC — This option affects input parity checking only, output parity generation is controlled by FCONTROL 36.

ADCC — This options controls both input parity checking and output parity generation, FCONTROL 36 only specifies the type of parity.

FCONTROL – 25 – Define alternate line terminator
This option is used to select an alternate character that will terminate terminal input in addition to carriage return. It is useful if your device terminates input with something other than return.

ATC — Backspace, linefeed, carriage return, DC1, DC3, Control-X, Control-Y, NULL, and DEL are not allowed as terminators. The manual claims that DC2 and ESC are not allowed as terminators but they work. If a DC2 is the first input character from an HP termtype terminal the HP3000 drops the DC2 and sends a DC1 back to the terminal, it thinks a block mode transfer is starting. Any other DC2 is recognized as a terminator if enabled. By enabling user block mode transfers (FCONTROL 29) a DC2 as the first character will also be recognized as a terminator when enabled. For non-HP termtype terminals a DC2 is always recognized as a terminator when enabled.

ADCC — Backspace, linefeed, carriage return, Control-X, Control-Y, and NULL are not allowed as terminators. The manual claims that DC1, DC3, ESC, and DEL are not allowed as terminators, but they work. DC2 is allowed as a terminator but produces bizarre results unless unedited terminal mode (FCONTROL 41) is also enabled in which case the DC2 is recognized as a terminator in any position.

If a line terminates with an alternate terminator, it will be included in the input buffer and length and an error will be indicated for the read. You must call FCHECK to determine that the read terminated with the alternate character.

```
fcontrol'parm:=[8/0,8/"."];  <<** period is alternate terminator **>>
fcontrol(pfnum,25,fcontrol'parm);
fs'error'on'ccl(pfnum,2513);
...
ilen:=fread(pfnum,ibuff',precsize);
if <
  then begin
    fcheck(pfnum,errorcode);
    if errorcode <> 31
      then file'error(pfnum,errorcode*100+2);  <<** something else **>>
    <<** handle alternate terminator **>>
  end;
```

FCONTROL – 26, 27 – Disable/enable binary transfers

Binary transfers can be used to transmit full 8-bit characters to and from the terminal. On input a read will only be satisfied by inputting all characters requested, a carriage return or alternate terminator will not terminate the read. No cr/lf is echoed to the terminal at the end of the read. Thus, you must always know how many characters to read on each input from the terminal. Enabling binary transfers also turns off the ENQ/ACK flow control protocol and carriage control on output. No special characters are recognized on input. See the note under FCONTROL 25 about DC2 as the first input character on a line. If a session device is being accessed in binary mode, a break will remove the terminal from binary mode but it will not be returned to binary mode when a resume is executed.

FCONTROL – 28, 29 – Disable/enable user block mode transfers

As described above the normal sequence of events in a block mode transfer from an HP terminal to the 3000 is for the HP3000 to send a DC1 to the terminal indicating it readiness to accept data, the terminal sends a DC2 when the enter key is struck to indicate that it is ready to send data, the HP3000 responds with another DC1 when it is really ready to take the data, and the terminal sends the data. All of this is transparent to your program which just issues a big read. If your would like to participate in this handshake you enable user block mode transfers and MPE relinquishes control of the handshake. Your program would issue a small read, get the DC2, and issue another read to accept the data. This allows you to meddle around before the data shows up.

The terminal driver only supports block mode transfers with HP termtypes and performs one other function during block mode transfers. Normally you wouldn't put a timeout (FCONTROL 4) on a block mode read because the user can take an indefinite amount of time to fill a screen; but you would like to avoid terminal hangs because the block terminator from the terminal gets lost. This situation is handled by the driver for you, the portion of the read after the second DC1 is sent to the terminal is timed for (#chars in read/#chars per sec)+30 seconds. If the terminator is lost and the read times out, the read will fail and FCHECK will return error 27.

```
fcontrol(pfnum,29,fcontrol'parm);
fs'error'on'ccl(pfnum,2913);
...
ilen:=fread(pfnum,ibuff',-1);
fs'error'on'ccne(pfnum,2);
<<** meddle/muddle **>>
ilen:=fread(pfnum,ibuff',precsize);
fs'error'on'ccne(pfnum,2);
```

FCONTROL – 30, 31 – Disable/enable V/3000 driver control

This option is an undocumented option in which the terminal driver provides low level support for V/3000 use of terminals. When V/3000 issues a read to the terminal the driver outputs a DC1; the terminal user hits enter which causes a DC2 to be sent to the 3000; the driver responds with ESC c ESC H DC1 which locks the keyboard and homes the cursor; it appears that the driver also enables binary transfers because the second read only terminates by count, not by terminator. The portion of the read following the second DC1 is timed as described under FCONTROL 28, 29.

FCONTROL – 34, 35 – Disable/enable line deletion echo suppression

This option suppresses the !!!cr/lf echo whenever a Control-X is received from the terminal, the Control-X still deletes all data in the input buffer.

FCONTROL – 36 – Set parity

This FCONTROL option sets the sense of the parity generated on output and checked on input. The four possibilities are: 0, no parity, all 8 bits of the data are passed thru; 1, no parity, the parity bit is always set to one; 2, even/odd, even parity is generated if the original parity bit of the data was a zero, otherwise odd parity is generated; and 3, odd parity, odd parity is generated on all characters.

ATC — FCONTROL 36 sets the parity sense and enables output parity generation. FCONTROL 24 must be called to enable parity checking on input. An undocumented effect of this FCONTROL call is that the previous parity setting is returned in the controlcode parameter wiping out its original value!

ADCC — FCONTROL 36 sets the parity sense only. FCONTROL 24 must be called to enable output parity generation which results in input parity checking as well. An undocumented effect of this option is that the previous parity setting is returned in the controlcode parameter wiping out its original value!

Parity is not reset to the default case when a device is closed. This can be useful if you have a session device that can not run with the default parity. Each time the system is started run a program that opens the device, sets the parity, and closes the device. It can then be accessed as a session device with the required parity.

ATC — The following results were obtained when parity generation was enabled on output. All options performed as described in the manual.

ADCC — The following results were obtained when parity generation was enabled on output. Option 0, parity pass thru, resulted in even parity on all characters. Option 1, parity forced to one, resulted in odd parity on all characters. Option 2, even/odd parity, resulted in even parity on all characters regardless of the original parity bits of the characters. Option 3, odd parity, resulted in odd parity on all characters. Only option 3 performed as expected.

ATC, ADCC — The following results were recorded when parity checking was enabled on input. Option 0, parity pass thru, resulted in parity errors on all input except that with even parity. Option 1, parity forced to one, resulted in parity errors on all input except that with odd parity. Option 2, even/odd parity, resulted in parity errors on all input except that with even parity. Option 3, odd parity, resulted in parity errors on all input except that with odd parity. Options 2 and 3 performed as expected, options 0 and 1 did not. In all cases, parity bits are always set to zero before the data is passed to your program buffer.

HP has told me that the following is the parity story as of the C-delta version of MPE-IV.

ATC — Options 0 and 1 will not check parity on input, everything else as described above.

ADCC — Option 0 and 1 will be parity pass thru, everything else as described above.

FCONTROL – 37 – Allocate a terminal

In the old days you had to allocate a programmatic terminal before it could be used. Now you don't even though the manual claims that you do. This option is still useful because it allows you to set the termtype and terminal speed with one FCONTROL call. Common sense, mine at least, says to set termtype and speed each time a device is opened even if the proper values are configured in the i/o tables. Using this option allows use of a file equation redirecting the program to another device that might not be properly configured.

```
fcontrol'parm:=[11/speed,5/type];
fcontrol(pfnum,37,fcontrol'parm);
fs'error'on'ccl(pfnum,37);
```

FCONTROL – 38 – Set terminal type

This option allows you to set the terminal type, but use FCONTROL 37 and set type and speed all in one shot.

FCONTROL – 39 – Obtain terminal type information

Before changing the terminal type, get the current value and reset it when you are through.

FCONTROL – 40 – Obtain terminal output speed

Before changing the terminal speed, get the current value and reset it when you are through.

FCONTROL – 41 – Set unedited terminal mode

Unedited terminal mode is about the most useful FCONTROL option used to communicate with programmatic devices. It allows almost all control characters to pass through to the HP3000 but does not require reads of exact length as in binary transfers. Input will terminate on a carriage return or an alternate terminator if specified. The subsystem break character, replacing Control-Y, can also be specified.

ATC — Unedited terminal mode overrides input parity checking, no checking is performed and all input parity bits are set to zero. Output parity generation is performed normally.

ADCC — Unedited terminal mode processes parity in the same manner as edited mode, see the section on FREAD for an explanation.

Binary transfers enabled overrides unedited terminal mode enabled. If the input terminates with the end-of-record character or alternate terminator no cr/lf is sent to the terminal. If the input terminates by count a cr/lf is sent to the terminal unless an FSETMODE 4 has been done. Unedited mode does not turn off the ENQ/ACK flow control protocol on the ATC or ADCC. See the note under FCONTROL 25 about using DC2 as a terminator.

PTAPE

The manual describes PTAPE as the intrinsic to use to read paper tapes. (A fancy data-entry media that is becoming increasingly popular.) It can be used on the HP3000 to access devices that send up to 32767 characters all in one shot subject to a few limitations. The data must be record oriented with carriage returns between records, MPE will cut the data into 256 character records if there are no returns, and the whole mess must be terminated by a Control-Y. Certain buffering terminals allow you to fill their memory off-line, connect to a computer, and transmit all the data. This could save considerable time and money over dial-up phone lines.

DEBUGGING

If you have a requirement to attach a programmatic device to the HP3000 the worst strategy is to write some code on the 3000, plug the device in and start testing. Murphy says it won't work and it won't. The method I use is to test the device, then the code, and then the code and device together. I test the device by plugging it into an HP-2645 (or equivalent) terminal, turning on monitor mode, and simulate the HP3000 by typing on the keyboard. (Remember that you are hooking two terminals together, you will probably hook device pin 2 to 2645 pin 3, device pin 3 to 2645 pin 2, and device pin 7 to 2645 pin 7.) You can stimulate the device and observe all responses quite simply. Any strange behavior can be noted at this point. The next step is to write the code on the HP3000 to access the device in the manner determined by the first tests. Then plug the HP-2645, not the device, into the HP3000. Now type on the 2645 to simulate the device, continue until your code is debugged. Now you can plug the device into the HP3000 and you have a good (modulo Murphy) chance of actually getting it to work.

REFERENCES

Communications Handbook, Hewlett-Packard Company, April 1981 Part #30000-90105. This manual supersedes the HP Guidebook to Data Communications and the Data Communications Pocket Guide.

Don Van Pernis, "HP3000 Series II Asynchronous Terminal Controller Specifications," Computer Systems Communicator, #15, December 1977, page 2. This explains the terminal subtypes and signal requirements for the ATC.

Charles J. Villa, Jr., "Asynchronous Communications Protocols," Journal of the HP General Systems Users Group, Volume 1, #6, March/April 1978, page 2. Good introductory material.

John Beckett, "Poor Man's Multidrop," Journal of the HP General Systems Users Group, Volume 2, #1, May/June 1978, page 7. How to hook several terminals to the same port.

Tom Harbron, "Lightning, Transients and the RS-232 Interface," Journal of the HP General Systems Users Group, Volume III, #3, Third Quarter 1980, page 14. How to avoid being zapped.

MPE Intrinsics Manual, Hewlett-Packard Company, January 1981 Part #30000-90010. Chapter 5 discusses most of the FCONTROL options that are applicable in terms of the ATC, it is often inaccurate in describing the ADCC.