A Brief Manual for QCTerm

QCTerm is not constructed as a precise mimic of either *Reflection* or even an HP700/92 terminal, although it identifies itself as the latter. Rather, we wanted to make *QCTerm* simpler, more browser-like, and more intuitive, while retaining the full functionality that would be expected of an HP terminal — eventually becoming fully competitive with the other PC-based terminal emulators.

Our hope is that we can make *QCTerm* sufficiently self-explanatory so that not all that much of a manual will be necessary. Whatever documentation proves necessary will be put here on the web.

Table of Contents

Some Features (& Tricks) of QCTerm

- The Modifier Keys
- The size of the Display Screen
- The 132-column Display is Undistorted
- The Arrow Keys
- The Page Up/Page Down Keys
- The Home/End Keys
- Clear Screen Function
- The Insert/Delete Keys
- Inserting and Deleting Lines
- The Pause/Break Key
- The Scroll Lock Key
- The Print Scrn/SysRq Key
- The Tab Key
- The Function Keys, F1 thru F12
- The CTRL+Function Keys, CTRL+F1 thru CTRL+F12
- The CTRL+Alpha Keys
- The CTRL+Alpha Keys (Transparent Mode)
- The Screen Buttons: Size, Color

Connectivity

- Current Options

Editing Functions

- Full-Screen Editing
- Finding and Replacing Text
- Selecting, Cutting, Copying and Pasting Text

Using & Programming the Function Keys

- Programming the Function Keys
- Function Key Attributes
- Selecting the User-Defined Function/System Key

Displays
Block Mode vs. Character Mode
Block Transfer Handshaking
Performing Block Transfers

Programming the Redefinable Keys

Programming the Cursor Keys

Extended Characters

Using Extended Characters

Autolaunch Scripting

Basic Design
Handshaking
Script Commands
Notes on Launching from a Browser

Some Features (& Tricks) of QCTerm

The Modifier Keys

For those of you who have extensively used WRQ's *Reflection* terminal emulator in the past, some adjustments will be necessary. Very few of the *Reflection* key sequences are used in *QCTerm*. Instead, we've attempted to make *QCTerm*'s sequences be in agreement with the standard sequences that have recently come to PC-based software.

In *QCTerm*, the ALT key is used **only** for the pull-down menu at the top of the screen. It is not used to generate any other form of *QCTerm* command sequence — with the exception of modifying the arrow keys and the break key, as noted below.

Rather, the CTRL key is used as the primary modifier for all other purposes than the pull-down menu. You'll see these differences as you read through the text below.

The Size of the Display Screen

The displayable screen size for *QCTerm* may be set in the "Terminal Preferences..." option of the "Terminal" pull-down menu to be either:

- 200 columns
- 132 columns
- 80 columns

The number of rows is fixed at 15,000.

In addition, there are potentially four font sizes for you to choose among. The largest font size is limited by your current screen resolution setting. The actual number of characters that will appear on your screen will depend on your PC's

screen resolution. At 640x480, you will only have one font size available to you. At 600x800, there will be two. At 768x1024, there will be three, and at 1200x1600 and higher, you will have four font sizes to choose among.

While *QCTerm* will work on PCs set to the lowest screen resolution, 640x480, we recommend that you use at least 600x800, if at all possible. Everything about *QCTerm*'s design was formatted so that it would look best and text would appear most appropriately when set to a 600x800 (SVGA) resolution, using Windows' Small Fonts option.

The 132-column Display is Undistorted

Rather than having the fonts become quite tall and thin when a 132-column display is selected, we elected to have *QCTerm* compress its screen height and drop to a smaller font size instead. Doing this allows the screen to maintain its original proportions.

The font-distorting "feature" of the 132-column mode on a regular HP terminal is a necessary consequence of the constraints that govern the terminal's design. However, on a PC, those constraints disappear and we elected not to mimic them.

The ←, , →, ↓ Arrow Keys

The arrow keys work as you would expect while you remain within the boundaries of the display screen. However, unlike a terminal (or *Reflection*), when you reach the top or bottom of the display screen, *QCTerm*'s cursor does not flip to the opposite side of the screen and continue its motion there. Rather, the terminal begins to automatically scroll, as if it were a web browser, three rows (or three character columns) at a time, depending on which way you're moving.

Each of the arrow keys can also be pressed with a modifer (SHIFT, CTRL, or ALT) key, somewhat akin to *Reflection*'s behavior. If the arrow key is pressed in conjunction with one of the modifier keys, the screen immediately begins to scroll, leaving the cursor in its screen-relative position, without having to cause the cursor reach one of the screen edges. Moreover, the three different modifier keys cause the scroll to occur at differing rates:

ALT causes the screen to scroll 1 column (or row) at a time.

CTRL causes the screen to scroll 3 columns (or rows) at a time.

SHIFT causes the screen to scroll 10 columns (or rows) at a time.

The modifier keys were designed in such a manner so that you're left with the feeling of increasingly putting "the pedal to the metal," accelerating the rate at which you scroll, but leaving you with an easy mechanism to choose the rate at which you scroll.

If you wish to use these high-speed scroll features to their fullest, you may wish to go to your Windows *Control Panel*, select "Keyboard," and set the "Repeat delay" to its shortest possible setting and the "Repeat rate" to its fastest possible setting.

The Page Up/Page Down Keys

The *Page Up*/*Down* keys cause the display to scroll up or down, 24 rows at a time. Although no modifier key is necessary to engage their behavior,

pressing the CTRL key does no harm. The CTRL key modifier is supported only for psychological compatibility with the *Reflection* terminal emulator.

The Home/End Keys

The *Home*/*End* keys cause the display to go to either its first screen row or its last screen row. As with the *Page Up*/*Page Down* keys, although no modifier key is necessary, simultaneously pressing the CTRL key will not modify their behavior. The CTRL key modifier is supported for psychological compatibility with *Reflection*.

Similar to *Reflection*, SHIFT+Home causes the cursor to return to the beginning of the current row while SHIFT+End places the cursor at the end of text on the current row.

Clear Screen Function

Although there is no *Clear Screen* key on a PC's keyboard, pressing the CTRL+Home key combination twice rapidly (within a quarter second) will cause the terminal to perform a **screen home and clear**.

Further, the key sequence CNTL+J will clear the screen from the cursor's current position to the end of display. Similarly, CNTL+K will clear the line from the cursor's current position to the end of the line.

These two sequences were chosen because they are similar to HP's standard escape sequences for "clear display," and "clear line": Esc J and Esc K, but in contrast to the escape sequences, Esc J and Esc K, the CNTL+J, CNTL+K keystrokes do not transmit anything to the host computer. They're acted upon wholly locally in *QCTerm* directly.

The Insert/Delete Keys

The *Insert* key puts the terminal into character insert mode. This mode is indicated by both an annuciator message at the bottom of *QCTerm*'s window and by a change in the appearance of the cursor. If the terminal is in its normal mode, with an underline cursor, the insert cursor will be indicated by a block cursor. If the terminal cursor is reversed, the Insert Mode cursor will similarly be reversed.

The terminal will remain in Insert Mode until (i) the *Insert* key is pressed again, (ii) an appropriate escape sequence is received, or (iii) the terminal is reset.

The *Delete* key takes effect immediately, deleting the character at the cursor's current location.

Inserting and Deleting Lines

The *Insert* and *Delete* keys' functions are modified with the CTRL key. If pressed simultaneously with the *Insert* or *Delete* keys, the keys become **Row Insert** and **Row Delete**, respectively.

The Pause/Break Key

If the *Pause-Break* key is pressed, an Xoff (CTRL-S) character is sent to the terminal, stopping the flow of data from the host computer to *QCTerm*. If the key is pressed again, an Xon (CTRL-Q) character is sent, resuming the data flow. The *Pause* button on the lower edge of *QCTerm*'s window acts both as an annuciator, declaring the current state of the pause, and as somewhat of a

twin to the *Pause-Break* key.

When the *Pause-Break* key is pressed with SHIFT modifier key, a BREAK signal is transmitted to the host computer. A BREAK may also be generated by pressing ALT+B.

ALT+B = *Break*
Shift+Pause = *Break*

The Scroll Lock Key

The *Scroll Lock* key is intercepted by Windows 95/NT at a processing level lower than *QCTerm*. It preforms a process something akin to Memory Lock in the terminal emulator, but not in a manner that is particularly useful. In general, it is recommended that you not use this key.

The Print Scrn/SysRq Key

The *Print Scrn* key is a second key that is intercepted in Windows 95/NT at a processing level lower than *QCTerm*, but in this case, the key's function can be very valuable for documentation purposes. Whenever the key is pressed, a bit-map snapshot of the entire current Windows display is put into the clipboard.

If Alt+*Print Scrn* is pressed, only the active window is placed into the clipboard. If *QCTerm* is the active window, a bit-map of the current screen, including high-lighted text and current function key settings, is made immediately available for pasting into other document-processing programs.

The Tab Key

When pressed as an unmodified key, the *Tab* key generates a horizontal tab character. When modified with the SHIFT modifier key, the *Tab* key generates a horizontal backtab character.

The Function Keys, F1 thru F12

The function keys F1 through F8 act in the normal manner appropriate to an HP Terminal. The remaining four function keys, F9 through F12, are assigned specific, non-modifiable functions. They are:

F9

Extended Characters. When pressed, the F9 key puts the terminal into an *Extended Character* mode, supporting either the extended ISO Latin-1 character set that Microsoft supports or the Roman 8/9 character set of Hewlett Packard. The choice of these character sets is made in the Terminal/Preferences... pulldown menu.

F10

User Keys. Pressing F10 will cause the terminal's user-definable function keys to be displayed at the bottom of the screen.

F11

System Keys. Pressing F11 will cause the topmost level of the terminal's System function keys to be displayed at the bottom of the screen.

F12

Enter Key. F12 is the ENTER key, as defined by a standard HP Terminal, *not* the RETURN key, which is often confusingly labeled as "Enter" on a PC's keyboard. Additional HP ENTER keys may be defined in the Terminal/Preferences... pulldown menu, including a SMART ENTER mode such that when *QCTerm* is block mode, the RETURN key automatically assumes the HP ENTER function.

If your screen display is wide enough, you will be able to see all twelve function keys. But even if the screen size is too small and these four extra function keys are not visible, it still will not be necessary to memorize their functions. They are recapitulated in *QCTerm*'s "Terminal" pull-down menu.

The CTRL+Function Keys, CTRL+F1 thru CTRL+F12

The function keys F1 through F12, when pressed with the CTRL modifier key, engage specific terminal display functions, allowing you to change the appearance of your screen "on the fly." The first four function keys control screen size. The next four control the font. The final four control the screen's color. You do not need to memorize these functions. They are repeated in *QCTerm*'s "Screen" pull-down menu. They are:

CTRL+

F1

Increases font size.

F2

Decreases font size.

F4

Maximizes/Minimizes Display, as a toggle.

F9

Inverts Color Motif, as a toggle.

F11

Previous Screen Color Motif.

F12

Next Screen Color Motif.

The CTRL+Alpha Keys

Certain CTRL+Alpha key combinations have become (or are on their way to becoming) standard on a PC. *QCTerm* enthusiastically supports the more standard of these combinations, simply because they greatly facilitate the ease at which one can move from program to program. These same key combinations are also supported by a growing number of software titles, including Netscape's *Navigator*, Microsoft's *Excel, Word, Internet Explorer* and *Visual Basic*, and America Online, among others.

These particular key sequences were originally defined in the early 1980's by Apple for the Macintosh. On that machine, Apple used a modifier key, called COMMAND, which unfortunately doesn't exist on a PC's keyboard. To compensate, people have used the CTRL key as the substitute modifier key when they've migrated these key combinations over onto the PC.

Among the more common of these sequences are:

CTRL+P = *Print Current Screen*
CTRL+S = *Save Current Work*

CTRL+A = *Select All*
CTRL+X = *Cut*
CTRL+C = *Copy*
CTRL+V = *Paste*
CTRL+F = *Find*

CTRL+Q = *Quit Program*

Some of these key sequences, however, clearly interfere with some of the same sequences that have been traditionally used on terminals, most notably CTRL+S, CTRL+Q, CTRL+C and CTRL+A.

The CTRL+Alpha Keys (Transparent Mode)

If you wish to use *QCTerm* as a traditional terminal emulator, a mode switch has been put into *QCTerm* that allows CTRL+Alpha characters to be passed on to the host and not processed locally. Two sequences have been defined to allow you to transmit any control key sequence to the HP3000. Once typed, *QCTerm* switches to Transparent Control Code mode. This mode is indicated to the user by having the text written in to communications control panel at the bottom of *QCTerm's* screen become bright blue.

To switch back to the newer user mode, use one of the two sequences shown below. Once done, the communications control panel returns to black lettering.

CTRL+T = *Switch to Transparent Control Code mode*
CTRL+Shift+F12 = *Switch back to Local Process mode*

ALT+T T = *Switch to Transparent Control Code mode*
ALT+T L = *Switch back to Local Process mode*

Some HP3000 sequences are common to both modes, however. They are:

CTRL+Y = *Subsystem Break*
CTRL+N = *Line drawing set on*
CTRL+O = *Line drawing set off*
CTRL+G = *Bell*

The CTRL+S and CTRL+Q (Xon/Xoff) pair of sequences, which act as a traditional data flow start/stop method, are transmitted to the host HP3000 only when in the CTRL+T mode. However, two additional flow control keys are present in *QCTerm* and they operate in either mode. They are the *Pause-Break* key on the keyboard and the *Pause* button at the bottom of *QCTerm*'s screen.

The Screen Buttons: Size, Color

Both attributes of the screen display (font size and color) can be set independently, "on the fly," while *QCTerm* is receiving data, either by pressing the appropriate CNTL+function keys or the on-screen buttons at the bottom of QCTerm's display.

The two on-screen buttons, *Size* and *Color*, cause their respective attributes to advance by one, instantaneously changing the screen size or color.

Similarly, both buttons can be caused to decrement their respective indexes if the buttons are pressed simultaneously while holding down any of the modifier keys (SHIFT, CTRL, or ALT).

Connectivity

Current Options

Two connection methods are available in the current version of *QCTerm*. They are:

- Serial Port Communication (to 256,000 baud)
- Telnet TCP/IP Communication (Standard & Advanced)

A more complete explanation of these connection options and how they work is available by clicking on the link. Serial and network connectivity mechanisms are more alike than you might originally imagine.

Editing Functions

Full-Screen Editing

The intention in *QCTerm* is provide a very powerful, but very simple full-screen (15,000 line) text editor. Every HP terminal has some very basic but very powerful full-screen editing functions built into the terminal functions, and when used with appropriate software on the host, the "LINE MODIFY" and "MODIFY ALL" keys provide an exceptionally powerful full-screen editor.

To add to these intrinsic capabilities, the standard text editing functions of find, replace, cut, copy and paste have also been added to *QCTerm*.

Finding and Replacing Text

To find a text string in display memory, beginning at your current cursor location, type either:

CTRL+F or ALT+E F *(find)*

to bring up a find & replace string definition window. You may request that the search be case sensitive by clicking the appropriate checkbox. The default is non-case sensitive. Press OK once the text has been defined to engage the find operation.

To find the next instance of the same string further down in display memory, type either:

CTRL+D or ALT+E D *(do it again)*

Found text will be marked as "selected", using an inverse box. Text that is selected may be either deleted by pressing the BACKSPACE key, or deleted character-by-character using the DELETE key, or it may be replaced with whatever phrase you earlier defined in the find windown by pressing either:

CTRL+R or ALT+E R *(replace)*

The delete and replace operations do not transmit either the individual characters or the line to the host. The operation is screen-only. If you want to transmit the modified line to the host, press either the ENTER (HP's ENTER function) key or the CARRIAGE RETURN, *if* the terminal is in "LINE MODIFY" or "MODIFY ALL" mode.

Selecting, Cutting, Copying and Pasting Text

You may either select text through the use of the FIND mechanism described above, or you may manually select text on *QCTerm*'s display memory using the mouse, using a standard left-click dragging operation.

To place the selected text into the PC's clipboard, you have several options. You may type either:

CTRL+X or ALT+E T *(cut)*

CTRL+C or ALT+E C *(copy)*

or you may simply right-click your mouse. The right-click operation will place the selected text into the clipboard, as a copy process. The successful completion of that operation will be indicated with the select being cleared.

You may also select blocks of text, not full lines, beginning a chosen row and column start point and ending at a similarly chosen row and column ending point. To do this, press and hold down the CTRL key *before* you left-click your mouse. Drag the mouse to the chosen endpoint, release the left button, and right-click your mouse to absorb the selected text into the clipboard.

The paste operation is invoked by first placing your cursor where you want to paste the clipboard text and then typing either:

CTRL+V or ALT+E P *(paste)*

or by right-clicking your mouse (when no text is selected).

The paste operation does not clear the clipboard, thus you may paste the clipboard text as often as you wish. Again, the paste operation is screen-only, similar to the replace operation described above.

Using & Programming the Function Keys

Programming the Function Keys

Eight programmable functions keys, F1 through F8 exist in QCTerm. Two methods exist to specify their content. The first is through the transmission of escape sequences from the host (please see the list of escape sequences). The second is by manually specifying the content of each function key through a screen that can be reached via the "Define Function Keys..." entry in the "Terminal" pulldown menu.

Function Key Attributes

A user-defined function key has one of three attributes:

Normal Mode

The user-specified key string is treated exactly as if it were typed from the keyboard. Of importance, a carriage return is **not** automatically transmitted at the end of the string. If you wish to have a carriage return be a part of the key string, put it into the string yourself. You may do that with the "Type in control characters" box checked. The same is true for a line feed character.

If QCTerm is in local mode, the string displays on the screen and any embedded escape sequences are executed locally. In remote mode with local echo off, the string is transmitted to the host computer. It executes and displays in QCTerm only if the host system echoes the string back to QCTerm.

"Normal mode" is the function key mode you'll want to use with most computer operating systems. "Transmit mode" was designed to work specifically with HP3000 computers running the MPE operating system. MPE provides an unsuppressable host prompt character, a DC1, at the beginning of each terminal read, essentially saying that "the host is ready to receive input." This character is used as a fundamental part of the handshaking protocol used in "Transmit mode" function keys, as explained below. Because this prompt is absent in UNIX, Linux, AS/400 and other operating systems, you will find that you will not generally want to employ "Transmit mode" function keys except under highly specific conditions, as outlined below.

In "Normal mode", each character in the function key string is transmitted individually, character-by-character. When connected to a host through a serial connection (e.g., RS-232), the effect is invisible. All characters are transmitted individually in this connection mode. However, when connected via telnet, the process is less efficient than under "Transmit mode", where the function key string is transmitted as a single packet. Unfortunately, to maintain functional compatibility with legacy applications written to the "Normal mode" of operation, the single-character nature of the transmission of the function key string cannot now be changed.

Transmit Mode

Under "Transmit mode", QCTerm transmits the user-specified key string to the host after completing a block transfer handshake (see below), automatically appending a carriage return (and a LF,

if the terminal is in AutoLF mode) to the string.

If a "Transmit mode" function key is pressed twice, before the necessary handshake associated with the first pressed key has not completed, the keyboard will momentarily lock as indicated by a small red square in the bottom-left corner of QCTerm's display screen. The lock will clear as soon as the data transfer and handshake of the first key have been completed.

In local mode, pressing a user-defined function key with the "Transmit" attribute has no effect.

Local Mode

The user-specified function key string is executed locally and not transmitted to the host.

Selecting the User-Defined Function/System Key Displays

Pressing QCTerm's F10 function key will always switch the displayed function keys to the User-defined set. If the function keys are not currently visible, pressing the F10 key will make them visible.

Similarly, pressing the F11 key will present the System function keys, making them visible if need be.

Block Mode vs. Character Mode

QCTerm is always in either block mode or character mode.

In character mode, characters are transmitted to the host one character at a time as they are typed. When the host receives each character, it echoes the character back to you and you see the character on the screen, although this process is slightly modified when typing in "advanced telnet" mode (see: connectivity options). ASCII control codes, such as CR and LF, and escape sequences, are also transmitted as they are typed.

When in block mode, QCTerm transmits a block of data at a time, thus the name. The size of each block can vary from just a few characters to several screen pages of data. ASCII control characters such as CR and LF work locally but are usually not transmitted with the block of data.

Block mode is engaged in one of three ways:

- A program running on the host computer activates block mode with the escape sequence Esc &k1B.
- You enter the block mode sequence from the keyboard manually.
- You toggle the BLOCK MODE function key, F3.

The keyboard locks automatically during a block transfer. A hard or soft reset unlocks the keyboard.

Block Transfer Handshaking

In general, the Inhibit Handshake (also called the "G strap", a name inherited from the days of the first terminals where the setting was made by physically moving a strap; now set by the escape sequences Esc &s0G and Esc &s1G)

and the Inhibit DC2 (the "H strap"; now set by the escape sequences Esc &s0G and Esc &s1H) determine the type of handshaking performed when in block mode. There are three types of handshaking:

None

With no handshaking specified, QCTerm simply transmits the block of data.

DC1

QCTerm transmits the block of data after receiving the DC1 character from the host.

DC1/DC2/DC1

Following the reception of a DC1 character from the host to trigger the transfer, QCTerm transmits a DC2 character back to the host to indicate that it's ready to transfer data. The host then sends a second DC1 character to signal that it's ready to receive the block, and the block is transferred.

Performing Block Transfers

There are four different methods to perform a block transfer:

Mode 1: Enter Key

Any of the ENTER keys (F12 or any of the other user-definable keys) is pressed to begin a block transfer.

The table that follows summarizes the transfer handshaking procedures based on the mode and G & H strap settings:

Mode	Inhibit Handshake (G strap)	Inhibit DC2 (H strap)	Handshake
Character	Off	Off	*None*
Block	Off	Off	*DC1/DC2/DC1*
Either	Off	On	*None*
Either	On	Off	*DC1/DC2/DC1*
Either	On	On	*None*

Mode 2: Esc d or status data request

This form of request begins when QCTerm receives an Esc d sequence from the host or a status data request is received. The status request can be a primary or secondary status request, a device status request, a cursor position sensing request, a command completion status (S, F or U) request, a terminal ID request, or a terminal features request.

The table that follows summarizes the transfer handshaking procedures based on the mode and G & H strap settings:

Mode	Inhibit	Inhibit DC2	Handshake

	Handshake (G strap)	(H strap)	
Either	Off	Off or On	*DC1*
Either	On	Off	*DC1/DC2/DC1*
Either	On	On	*None*

Mode 3: Transmit Function Key

A user-defined function key that is configured in the "Transmit mode" (as explained above) is pressed.

The table that follows summarizes the transfer handshaking procedures based on the mode and G & H strap settings:

Mode	Inhibit Handshake (G strap)	Inhibit DC2 (H strap)	Handshake
Character	Off	Off	*DC1*
Block/Line	Off	Off	*DC1*
Block/Page	Off	Off	*DC1/DC2/DC1*
Character	Off	On	*DC1*
Block/Line	Off	On	*DC1*
Block/Page	Off	On	*None*
Any	On	Off	*DC1/DC2/DC1*
Any	On	On	*None*

Mode 4: MODIFY modes on and keypad ENTER key is pressed

The ENTER key is pressed when either the LINE MODIFY or MODIFY ALL keys are enabled.

The table that follows summarizes the transfer handshaking procedures based on the mode and G & H strap settings:

Mode	Inhibit Handshake (G strap)	Inhibit DC2 (H strap)	Handshake
Character	Off	Off or On	*None*
Block	Off or On	Off or On	*N/A*
Character	On	Off	*DC1/DC2/DC1*
Character	On	On	*None*

Programming the Redefinable Keys

Programming the Cursor Keys

Beginning with QCTerm Version 0.95, the ten keys on a PC keyboard that are labeled with cursor positioning labels may also be user defined. The keys are the INSERT, DELETE, HOME, END, PAGE UP, PAGE DOWN, UP ARROW, DOWN ARROW, LEFT ARROW, and RIGHT ARROW. The sequences to define these keys are:

```
Esc %f&ltx&gtk&lty&gts&ltl&gtL&lttext>
```

where

```
&ltx>
45 = insert key
46 = delete key
36 = home key
35 = end key
33 = page up
34 = page down
38 = up arrow
37 = left arrow
40 = down arrow
39 = right arrow

&lty> (optional)
0 = carriage return appended to string (default)
1 = carriage return at end of string suppressed
```

and

`&ltl>` is the length of the following text string.

As an example, to program the HOME key to transmit a sequence to the host, the desired sequence might be:

```
Esc %f36k8Lshowtime
```

These keys definitions only work in QCTerm. They are not part of the standard HP terminal escape sequence definition. The behavior of these programmable keys is a hybrid of the "normal" and "transmit" characteristics of the standard function keys.

In the absence of an explicit carriage return suppress command, a carriage return is appended to the sequence of characters you program, but pressing the key a second time doesn't wait for a host prompt (a DC1) to be returned before it transmits the text sequence again, as a "transmit" function key does. However, the text you specify is transmitted as a single packet, rather than a character at a time, as a "normal" function key operates.

To clear a programmed key, transmit a zero-length definition, in this manner:

```
Esc %f36k0L
```

A hard reset will clear all of the key definitions and reset the cursor keys to their normal terminal functions.

Extended Characters

Using Extended Characters

Several mechanisms may be used in *QCTerm* to engage the extended character set. They are:

- the use of a nationalized keyboard,
- the use of the ALT+numeric keypad (e.g., ALT+0226 = â), Microsoft's manner of generating a non-English character,
- the use of *QCTerm*'s F9 key, *Extended Characters.*

The F9 key method was designed to be a simple, high-speed mechanism to allow near-touch typing speeds, once learned. The F9 key is a mode switch, putting you in and out of the *Extended Character* mode.

If you press the F9 key just prior to striking the "u" key, the symbol displayed will be the "ú". If this is not the accent mark you wished, you may then strike the "u" key again, while still in the *Extended Character* mode, and a second accent mark will be displayed. This process will be repeated until you have cycled through all available accent marks or you strike some character other than a "u".

This cycling process has been defined for a number of keys on the keyboard, allowing you to create the entire ISO Latin-1 character set without having to memorize their positions or their numeric equivalencies. Keys were logically grouped together in as an intuitive a manner as possible.

You can see this cycling pattern in the following F9 key modifications:

The ISO Latin-1 character set:

á à â â ä ã å æ ª a
é è ê ë e
í ì î ï i
ó ò ô ö õ ø œ º o
ú ù û ü u
ý y
ç c
ñ n
š ß s
ð þ t
ž z
… · • .
¹ ¼ ½ 1
² 2
³ ¾ 3
‰ %
± +
× ° *
€ £ ¢ ¥ ¤ ƒ $
“ ” « » "
© ® ™ @
¡ !
¿ ?
÷ /

Once you have selected the accented character you wish, typing any character other than the one you began with will automatically disable the *Extended Character* mode and advance the cursor one space forward.

If you need to type two different extended characters in a row, after you have completed the first character, press F9 again. This will advance the cursor one space forward and leave you in the *Extended Character* mode.

The sequence of accented characters shown above is the sequence that would be preferred by an English, French or Spanish typist. Typists typing in other languages would have different preferences, and those too have been programmed into *QCTerm*. The choice of accent order by language preference may be selected in the Terminal/Preferences... pulldown menu.

The accent sequences shown above are for ISO Latin-1. *QCTerm* also supports HP's Roman 8/9 character set. The philosophy remains the same for this second character set, but because Roman 8/9 does not contain as extensive a character set, some characters in the list above will be missing.

While you are in the *Extended Character* mode, many of *QCTerm*'s keyboard functions are disabled, such as Page Up, Delete Line, etc. If you wish to execute these functions, you must first leave Extended Character.

Autolaunch Scripting

Basic Design

The intention underlying the design of the scripting structure in *QCTerm* was to constrain the number of commands to an absolute minimum and yet allow you the full flexibility to do anything. It was also the intention to eliminate any form of host dependency so that *QCTerm* will work with any host operating system (MPE, Linux, UNIX, MVS, OS/400, etc.).

QCTerm autoregisters itself into the Windows registry each time that it is run so that the file extension ".qct" is linked to *QCTerm*. The result is that if a file that has been placed on the user's desktop that has the qct extension is double-clicked, *QCTerm* will autoload and then commence executing the instructions contained within the file.

Quite similarly, if the same script file were placed on a web page as a hyperlink, as in this example:

```
<a href="http://aics-research.com/hal9000.qct">
```

the file, *hal9000.qct*, would be first downloaded into a temporary directory associated with your browser and then executed from there, in exactly the same manner as the desktop icon.

A simple launch script in QCTerm might be:

```
addr 67.41.4.238
wait \017
```

```
xmit hello myscript,demo.qcterm
wait \017
xmit basic
wait \017
xmit get hal9000
wait \017
xmit list
wait \017
head Sample Script
exit
```

To create an autolaunch *QCTerm* icon for your desktop, bring up any text editor (Microsoft's *Notepad* is quite excellent for this use) and type in a script very much like the one above (which will, in fact, work). Save the completed script with a ".qct" extension. Drag and drop the newly created file onto the desktop. At that point, you're done. It is no more complicated than that.

Handshaking

The core of the scripting mechanism in *QCTerm* revolves around the very simple idea of the WAIT/XMIT command pairs. The script waits for a particular string to appear in the terminal's receive buffer and then, when detected, transmits a response. The script then waits for another such sequence of characters, and then transmits a second response, and so on, repeating this sequence as often as necessary.

If the host runs an operating system such as MPE, the author of the script knows that the operating system will always prompt the terminal with a "go ahead" signal. In MPE, that host prompt is the DC1 character (\017 in decimal notation). However, many operating systems do not have or use specific host prompt characters. Rather, you must look for a specific string of characters in the received text that certify the end of a transmission.

Such a string might be "/$". A problem occurs if that specific string occurs prematurely in another context before the sequence that you're really seeking.

The cure is straightforward. Simply put two WAIT statements in a row, as in this example:

```
WAIT copyright
WAIT /$
XMIT list myprog
```

This particular choice of words was perhaps chosen by observing that the word "copyright" occurs in your host's text after the first appearance of "/$" and somewhat just prior to the real instance of "/$", the one that indicates the host is truly ready to receive data.

In this manner, you are assured that you are responding only at the appropriate time.

Script Commands

The primary script commands are:

- **ADDR** - ADDR or DIAL must be the first command. The ADDR command is intended only for network connections and is followed by either the DNS or IP address of the intended host. Although telnet's port 23 is presumed by default, a different port may be specified by

appending the port number to the IP or DNS address using either a colon or space syntax in this fashion:

```
addr 67.41.4.238:9423
addr 67.41.4.238 9423
```

- **DIAL** - ADDR or DIAL must be the first command. The DIAL command is intended only for modem connections and is followed by the phone number of the intended host, including any modem commands that might be necessary to dial out. The standard modem command "ATDT" is automatically appended to the beginning of the string in order to cause the modem to dial. Examples of reasonable strings are:

  ```
  dial 9,,1 505 555 1212
  dial 523 6782
  ```

  For more information concerning automated dialings, please see the IDLE command below.

- **WAIT** - waits for the specified text to appear in QCTerm's receive buffer. Non-printable characters may be specified by their decimal equivalents in an *nnn* format (e.g., \017 is the DC1 and \027 is the escape character).

- **XMIT** - transmits the specified sequence to the host computer, followed by an implicit carriage return (and LF, if AutoLF is enabled).

- **EXIT** - causes script execution to cease and relinquishes control to the terminal emulator.

The auxiliary commands are:

- **HEAD** - inserts the specified text into QCTerm's caption bar at the top of the screen and into its taskbar button.

- **LOCL** - executes the specified escape sequences locally, without having the sequences transmitted to the host. The standard terminal escape sequences are the mechanism used to control QCTerm.

  A sample local command sequence might be:

  ```
  locl \027H\027&a10r2C\027dB\027F
  ```

  which, when interpreted, would home the terminal to top-of-screen, set the cursor to the eleventh row, third column, invert the screen display for the remainder of the line, and then place the cursor at the bottom-of-screen.

- **IDLE** - causes the script to idle for the specified number of seconds, which may range from anything greater than 0 to 1000 seconds. The command is likely to find use only when dialing into a remote host using a modem. It's generally not necessary for network-based connections. Many modem-based remote-access servers do not become fully awake until a second or two after a connection has been made, thus scripts such as the following often prove necessary:

  ```
  dial 1 573 555 0738
  wait B
  idle 1.5
  xmit
  ```

```
wait :
exit
```

The script above results in the following screen display:

```
atv1q0
OK
atdt 1 573 555 0738
CONNECT 28800/ARQ/V34/LAPM/V42BIS

Welcome to USRobotics
The Intelligent Choice in Information Access
login:
```

A few notes on using scripts with a modem. Although one to many characters may be in QCTerm's receive buffer at any one time, with serial connections you cannot depend on a specific string to be there in its entirety as you can with a telnet connection, which transmits its material to the terminal as packets. Serial connections often transmit their strings as individual characters, thus the script above is looking only for a single-character string, the "B", to indicate that a connection has been made. Once the "B" has been detected, the script waits for 1.5 seconds for the remote access server to "wake up." An empty XMIT is then sent, which results in only a carriage return being sent to the server, the server's attention character. The script then waits for a colon to appear, indicating that the server is ready for the login name. Although not shown in the example above, a more complete dialog of password and host signons could easily be appended to the script.

- **DBUG** - has only two operands, ON and OFF. When debug is on, all transmissions during script negotiations (coming and going) are logged into the "c:\aics\temp\qcdebug.txt" text file so that you can easily see what's working and what's not.

  Thus, a more elaborate script might perhaps be:

```
addr 67.41.4.238
dbug on
wait \017
xmit hello myscript,demo.qcterm
wait \017
xmit basic
wait \017
xmit get hal9000
wait \017
xmit list
wait \017
locl \027H\027&a2R\027&dB
head Listing a File in New Mexico
dbug off
exit
```

Notes on Launching from a Browser

There are two ways that a browser launches an auxiliary, helper application. The older method, and the only process that Netscape's *Navigator* currently supports, is through the use of a MIME type.

The MIME type for *QCTerm* is:

```
application/x-qcterm
```

In order for this mechanism to work, the MIME type must be registered both in the PC and in the remote server. *QCTerm* autoregisters itself in Windows' Registry, but the MIME type must also be (for the moment) manually placed in the server's list of supported MIME type associations and equated to ".qct".

Microsoft's *Internet Explorer* 5.0 and above support not only the MIME-type registration mechanism, but also the far simpler file type extension. When *IE5* encounters a file extension in a hyperlink, it looks in its local registry to determine the nature of the application associated with the extension. If that association is registered, and the application exists on the PC, the application is autolaunched without reference to the MIME type.

This is a far simpler mechanism than the original MIME-type protocol that Netscape has chosen to continue to support. Nonetheless, if you are going to create an intranet or internet autolaunch application for *QCTerm* where Netscape users are likely, you must manually register *QCTerm*'s MIME-type association in your server(s).
