PowerTCP™ Read Me

Notes on PowerTCP

This file provides important information not included in the *PowerTCP Developer's Guide*.

This file is divided into the following sections:

1. Changes and Additions to the Manual

The RecvEvent() functions (Recv Event for VB) will be called with NULL arguments whenever the underlying communications socket is closed.

The VT220 Custom Control Text Property now has a CR/LF at the end of each line. In 80-col mode there will actually be 82 characters per line. This applies to the Styles property also.

ConnectUdp() function. Callback types on page 28 of volume 2 should be CONNECTUDPEVENT and EXCEPTIONUDPEVENT. This function will fail, returning a NULL, if ExceptionEvent is not specified.

ConnectEvent() function on page 30 of volume 2 is of type CONNECTUDPEVENT.

ExceptionEvent() functions on page 30 of volume 2 is of type EXCEPTIONUDPEVENT.

All TELNET and TCP DLLs and VBXs have been updated to version level 3. (ie. P16TCPB3.VBX). This was done to accommodate the addtion of a new function parameter. The Recv(size_t MaxRecvCnt) function now has this new parameter that can be set to 0 (to disable the receipt of data, thereby producing backpressure to the host) or any other number to limit the size of any data "chunk" that arrives on the port. This function can be used to pause a terminal emulator, for example. The PT_RECV flag is consequently no longer needed, and has been deleted.

The PT_TCPNODELAY flag has been added. This can be used to set the TCP_NODELAY option for the socket in use.

The VT220 Custom Control PrintPassthrough property has been added. When set, printer output goes directly to the printer, bypassing the usual device driver processing. This is useful for PostScript printer in PrinterController mode. The new name for the control is P16VT2B2.VBX.

2. Directory Structure

The list of directories is shown below, with a description of each directory:

\POWERTCP
readme files and documentation is put it the base directory
rfcs.zip includes many TELNET and FTP RFC documents

\BIN
Contains the required fonts, DLLs and VBXs if the user chose
to install them here. The user may have chosen
to install the files in the Windows' system directory.
Also contains all help files.

\INCLUDE
Contains all header files for C/C++ and the constants
file for Visual Basic.

\LIB
Contains .LIB files for C/C++.

\SAMPLES
Contains sample programs.

\C_DG
The source code for a simple UDP datagram program.

\C_ECHO
The source code for a simple ECHO program. Utilizes the P16TCPC2.DLL interface and
demonstrates both client and server functionality over the TCP interface.

\C_TELNET
The source code for 4 simple TELNET clients. Visual C++ "make" files are included for 16
and 32-bit applications that link to PxxTNTC2.DLL (C interface) and PxxTNTS.LIB (C++
interface). The same source code is used for both 16 and 32 bit versions.

\C_TLNT32
The 32-bit versions of the files in C_TELNET.

\C_FTP
The 16-bit version of an FTP application.

\C_FTP32
The 32-bit versions of the files in C_FTP.

\DLL
Source code used for the construction of the FTP, TCP and TELNET DLL libraries.
Provides insights into how both the C++ and C interfaces operate.

\VB_TALK
An example conversation program which uses TCP
to send messages back and forth between two
different computers. Users type on one computer
and their message will be received on another.
This is for Visual Basic.

\VB_TCPEX
The tutorial TCP program described in the manual
for Visual Basic

\VB_VT220
A typical TELNET client program in Visual Basic. Utilizes the new VT-220 control

\VB_TNTEX
The tutorial TELNET program described in the manual
for Visual Basic

\VB_FTP
Sample VB test program demonstrating FTP VBX

3. Running the Visual Basic Examples

To run the Visual Basic examples, you **must** have VBRUN300.DLL installed on your system (in your windows directory or windows\system directory). If you do not have this, it is available from countless places, including the Internet and online services such as *America Online* and *Compuserve*.

4. Maintenance Reports

If you encounter a problem while testing PowerTCP, please
describe the difficulty and notify about the following items:

- What were you doing when the bug occured?
- Can you consistently duplicate the bug?
- Can you run the test programs?
- Can you compile the test programs?
- What TCP/IP transport are you running on, and can the behavior be duplicated on other transport?

Send all reports to the address listed below ("How to reach Dart Communications") or to the following
e-mail address:

support@dart.com

5. Common Questions and Answers

This section provides answers to some common questions for Visual Basic users, although most answers apply to C and C++, also.

Q: When I use the MsgBox function in Visual Basic, the PowerTCP control does not receive any events while the message box is up. For example, if the other end of a connection closes when I have a message box up, I never receive the Exception event. Why is this happening, and how can I make sure I receive the events?
A: There is a limitation in Visual Basic which prevents events from being generated while a message box *or* an input box (InputBox$()) is on the screen. This limitation only exists with the Visual Basic functions MsgBox and InputBox$, however, and does not exist with the Windows API function MessageBox. The solution is to use the Windows API MessageBox function instead of the standard MsgBox function. See Section 8 for more information.

Q: What are all the **PT*xx*.DBG** files in my default directory?
A: If the **PT_DEBUG** flag has been set to enable debugging for PowerTCP, all sent and received data is captured into a single file. You may want to delete these files after each time you use debugging with PowerTCP, as some of them may be quite large, depending on the data sent and received.

Q: When I receive a carriage return in the **Recv** event and place it in a text box, it shows up

as a black square. How can I make it appear as a carriage return?
A: Carriage returns are sent as **Chr$(**13**)**. In Windows, ending a line requires *two* characters--a **Chr$(**13**)** followed by a linefeed character (**Chr$(**10**)**). To solve your problem, you must search through the data you receive for carriage returns and add a linefeed character after each one before you place them in a text box.

Q: The same thing happens when a backspace character is received—instead of deleting the previous character, it places a black box in the text box I add it to. How can I make it delete the previous character?
A: Backspace characters are send as **Chr$(**8**)**. You must search for these characters when you receive them, and perform the delete yourself when you find one. For example, if you receive the backspace character, instead of placing it in a text box, you should say *Text1* = **Left$(***Text1*, **Len(***Text1***)** - 1**)**. This will delete the last character from the text box.

6. How to Contact Dart Communications

You can reach Dart Communications in the following ways:

Address
61 Albany Street
Cazenovia, NY 13035
Phone
315.655.1024
Fax
315.655.1025
E-mail
support@dart.com

7. Compiler Options

You will note that Dart has included static libraries for the small, medium, and large models. We have also included P16TCPxD.LIB versions (for DLLs) of the same. If you are constructing an EXE, use the former. If you are constucting a DLL, use the latter. The former are compiled with the /GA flag, and the latter are compiled with the /GE flag. It all has to do with what value Windows loads into the DS register when our PowerTCP window is called.

If you are building an EXE, use the following options:
1. Assume DS==SS (do not use /ASu or /ASw)
2. If using the C interface, ensure all callbacks are marked as __export or use /GEf
3. Use Protected Mode Application Functions (/GA)
4. MakeProcInstance is not required for callbacks using VC++ 1.5 as the compiler takes care of the details.

If you are building a DLL, use the following options:
1. Do not assume DS==SS, DS NOT loaded on function entry (/ASw)
2. If using the C interface, ensure all callbacks are marked as __export or use /GEf
3. Use Protected Mode DLL functions (/GD)

Note on use of STRICT. PowerTCP was compiled with STRICT defined, meaning that it will only link correctly with other modules compiled with STRICT defined. If this is not possible, then replace the HINSTANCE type with STRICT version wherever it is used: void NEAR *, and cast your hInstance variable to this type. This should allow you to link successfully without unresolved external errors.