PowerLine Help Index

HELP ME!

Procedures

- How To
- Chapter Reference

Reference

- Menu Reference
- Scripts (PowerLine BASIC)
- PowerPad Reference
- Glossary

Help Instructions

You can find information in Help by using the Index or the Search feature. To return to the Help Index after you read the instructions below, click on the Back button at the top of the Help window or press the B key.

For detailed help on using Help, press the F1 key while this or any Help window is active.

To use the Help Index

Do one of the following:

- Click an underlined topic. On color displays, the underlined topic is green.
- Press the TAB key until you highlight the topic you want, and then press the ENTER key.

To scroll in the Help window

Do one of the following:

- Press the Up Arrow and Down Arrow keys.
- Click on the scroll bar arrows with the mouse.

To return to the previous topic

Click the Back Button or press the B key.

To open the Search feature in Help

Click the Search button or press the S key.

To close the Help window

From Help's File menu, choose Exit (ALT, F, X).

PowerLine Menu Reference

Click on any of the below menu names to see a list of available menu items:

File menu

Disk Capture
File Manager
Print Capture
Printer Setup
Exit

Edit menu

- Add To Scratch Pad
- View Scratch Pad
- Copy
- Paste
- Find Text
- Find Next
- Clear:
  - Screen
  - Scrollback
  - Both
- Copy Scrollback to:
  - Clipboard
  - Disk File
  - Printer
- Copy Screen to:
  - Clipboard
  - Disk File
  - Printer
  - Scrollback

Watch Us Grow!

This reference depends on the completion of the documentation, and is not finished. Sorry. IF YOU SEE THIS MESSAGE IN A PRODUCTION VERSION OF POWERLINE, PLEASE CONTACT THE DOCUMENTATION DEPARTMENT!

Setup menu

- Hardware
- Terminal
- Function Keys
- Character Map
  - Incoming
  - Outgoing
- File Paths
- Mouse
- Protocol
- Toolbar
- Preferences
- Dial Options
- Port/Modem

Connect menu

Phonebook
Manual Dial
Dial Current Listing
Send a File
Receive a File
PowerLink
Hangup Modem
Initialize Modem
Wait for Call
Send Break
Local Echo
Chat Window

Script menu

Run
Manager
Stop
Learn

Design menu

Menu
Toolbar

Help menu

Chapter Reference

Below is a list of chapters as they appear in the printed PowerLine manual. To see a detailed list of topics covered in the chapter, click on the name of the chapter:

Introduction

This chapter is available only in the printed documentation.

Appendices

With the exception of the glossary, the Appendices appear only in the printed documentation.

How To...

Advanced Features

Sample Scripts

This topic is available only in the printed documentation.

Getting Voice Help

If you don't like to read lengthy help descriptions, no problem, with your multimedia MS Windows compliant sound device you can access PowerLine's voice help or even create your own audible help messages. First, make sure you have installed the voice help files and activated the help by using the Settings Dialog box. Better yet, make and play your own sound files by using PowerLine's scripts or by replacing our help files. To access voice help just select Voice help from the Application Control icon on any main dialog box and listen.

Please note that, although you will be able to get sound from your PC speaker if you have installed a PC speaker driver, this capability was designed for a sound board (such as SoundBlaster 16, SoundBlaster, SoundBlaster Pro, ThunderBoard, or Pro Audio Spectrum), and you may experience a degradation of sound using only a PC speaker.

To create your own voice help messages, replace our help sound files (.wav) with one of your own by the same name. Use the following chart as a reference:

File Name	Dialog Box
V0.WAV	Disk Capture
V1.WAV	Find
V2.WAV	Hardware Setting
V3.WAV	Terminal Setting
V4.WAV	Phonebook Entry
V5.WAV	File Paths
V6.WAV	Mouse Setting
V8.WAV	Keyboard Map
V9.WAV	Filter Map
V10.WAV	Phonebook Manager
V11.WAV	Quick Connect
V12.WAV	Send File, Receive File
V13.WAV	PowerLink
V14.WAV	Run a Script
V15.WAV	Script Manager
V16.WAV	Scratch Pad
V17.WAV	Menu Editor
V18.WAV	Print Capture
V19.WAV	Save As
V21.WAV	System Settings
V22.WAV	Change Color
V23.WAV	Create a Phonebook
V24.WAV	Delete a Phonebook
V25.WAV	Common Protocol Setup
V26.WAV	ASCII Protocol Setup
V27.WAV	Zmodem Setup
V28.WAV	Dialing Setup
V29.WAV	Port/Modem Setup

PowerPad Help Index

HELP ME!

You'll find manipulating ASCII text easy with PowerPad, PowerLine's ASCII text editor. In addition to editing scripts, you'll find PowerPad useful when editing any ASCII files or when you need to create mail for your favorite information service.

You can access PowerPad from the PowerLine group icon or you can define a function key to access it by using PowerLine's keyboard map feature. The default keyboard map displays an Editor button which will access PowerPad. PowerPad also supports the standard MS Windows standards for selecting text and moving within a document.

- File menu
- Edit menu
- Search menu
- Help menu

PowerLine BASIC reference

File menu (PowerPad)

Item	Description
New	Create a new PowerPad file.
Open	Open an existing file.
Save	Save the file in the current window under its current name.
Save As	Save the file in the current window under a new name.
Save&Exit	Save the file in the current window under its current name and exit PowerPad.
Print	Print the current file.
Setup	Use the MS Windows Print Manager to setup your printer. See the MS Windows manual for more information.
Exit	Exit PowerPad.

Edit menu (PowerPad)

Item	Description
Undo	Undo the last edit you made.
Cut	Cut the selected text from the PowerPad file and place it on the Clipboard.
Copy	Place a copy of the selected text on the clipboard.
Paste	Paste the copy or cut text from the clipboard to the PowerPad document.
Delete	Delete the marked text.
Select All	Select all text in the current document.

Search menu (PowerPad)

Item	Description
Goto Line#	Move the cursor to the specified line number. This item is especially useful when editing scripts. When an error is found in a script PowerLine displays the line number that contains the error.

Help menu (PowerPad)

Item	Description
Index	Displays the help index.
Scripts	Provides help on PowerLine BASIC script commands and procedures.
About	Display the PowerPad description.

Disk Capture (File menu)

When you communicate with another computer, you may want to save some or all of the information that appears on your screen. Use PowerLine's Disk Capture feature to "capture" this information and save it on your data disk to an ASCII file. In addition to saving the information to a file, you can also send it to the printer. Save data in memory by using the Copy to Scrollback feature. See Print Capture and the Setup Menu, Preferences for details on scrollback.

Activate disk capture to store any data received to an ASCII file. Choose to include related characters or filter them. Capture to disk when you need to store large amounts of incoming data that might, if captured in memory, exceed the 150 page scrollback limit. You cannot view data captured to disk while disk capture is active.

If you are capturing data to a floppy disk, do not remove the disk from the drive until you have deactivated capture; a Windows error message displays if the disk is removed prematurely.

To capture data to disk:

- From the Top Line Menu, select File, Disk Capture. Respond to the prompts in the Disk Capture Manager dialog box.

Disk Capture Manager dialog box

Use the Disk Capture Manager dialog box to capture data to a new file or append data to an already existing file. Additionally, choose to filter transmitted characters after designating a Character Map from the Setup Top Line Menu option.

Item	Purpose
File Name	Select or enter the name of a file to which captured data will be saved. Enter a new name if the data is to be placed in a new file; enter the name of an existing file if the data is to overwrite the data in the existing file, or be appended to the data in the existing file. If the file contains data, and you did not choose to append the new data to the already existing data, the existing data will be overwritten. You will be prompted to accept or cancel the overwrite. If you do not enter an extension, PowerLine assigns .cap.
Directories	Select the appropriate directory.
List Files of Type	Display a list of files with the .cap extension, or all files from the designated path.
Drives	Select the appropriate drive.
Open	Begin data capture. The terminal screen redisplays. When Disk Capture is active, Capture File (Filename) displays in the Status Line.
Exit	Exit from the dialog box.
Pause	Pause the capture of incoming data.
Resume	Resume the capture of incoming data.
Close	Stop data capture; file to which you have been capturing data saves data and closes. When Disk Capture is closed, Capture Closed displays in the Status Line.
Delete	Delete the file specified in the Delete dialog box.
View	View and edit selected file in PowerLine's PowerPad.
Filtered	Activate the character map table; omit carriage returns, line feeds, control characters, escape sequences from the capture file.
Append	Append incoming data to existing data in the designated file.

File Manager (File menu)

Displays the Windows File Manager. See your Microsoft Windows manual for more information.

Print Capture (File menu)

To send the information scrolling on the screen to the printer, select the Print Capture option. Select Open to send the data displaying on the screen to a file that will be sent to the printer when you execute the Close option. The file no longer exists after the data is sent to the printer; the maximum size of the file is determined by the configuration of your Windows operating system. Additionally, you can interrupt and resume sending data to the printer.

To print data received from another computer:

- From the Top Line Menu, select File, Print Capture. Respond to the prompts in the Print Capture dialog box.

Print Capture dialog box

Item	Purpose
Printer	The active printer; to change printers, access the Windows Print Manager.
Device	The active port; to change ports, access the Windows Print Manager.
Copies	The number of copies that will be printed; to change the number; access Windows Printer Setup.
Filtered Output	Activate the filter mapping table to prevent control characters from being printed.
Open	Activate to send output to printer; printing begins when the Close option is selected.
Exit	Exit Print Capture dialog box.
Pause	Interrupt printing.
Resume	Resume printing at the point of interruption.
Close	Stop sending data to the print file; begin printing data from the printer file.

Printer Set (File menu)

Access the Print Setup dialog box to customize printing options for your installed printer or change the active printer. See your Microsoft Windows manual for more information.

Exit (File menu)

Select Exit to exit PowerLine.

Add To Scratch Pad (Edit menu)

Add marked information to the Scratch Pad.

To add information to the Scratch Pad:

1. Mark the appropriate information using the mouse.
2. Select Edit, Add to Scratch Pad. The information will be added to the Scratch Pad.

Copy (Edit menu)

Copy a segment of marked text to another location. The copied information will be held in memory and, if the Clipboard is active, placed on the Clipboard. To retrieve copied information select Paste from PowerLine or any Microsoft Windows application that supports the paste function.

To copy information:

1. Mark the appropriate information.
2. From the Top Line Menu, select Edit, Copy.

Paste (Edit menu)

Paste text from PowerLine or any Windows application that supports the edit function.

To paste:

- Move to the application or area where you want to paste information and from the Top Line Menu, select Edit, Paste.

Find Text (Edit menu)

Find a designated text string on-screen or in the scrollback.

To find text:
- From the Top Line Menu select Edit, Find. The Find Text dialog box displays.

Item	Purpose
Find Text	Enter text for which PowerLine should search. Only text in scrollback and on-screen is searched. Maximum length of text string is determined by screen column designation. See the Setup Menu, Preferences and Terminal for details on screen column designation.
Case Sensitive	Activate to find text string with identical upper/lower case designations.
Search Backward	Activate to search backward from cursor position to beginning of document. Deactivate to search from cursor position to end of document.
Find	Activates search.

Find Next (Edit menu)

Use Find Next to locate the next occurrence of the designated text string on-screen or in the scrollback.

To find the next occurrence:
- From the Top Line Menu, select Edit, Find Next.

Clear Screen (Edit menu)

Clear data from the screen. Note, that if your data has not been saved to disk or sent to the printer, it will be lost.

To clear all data from the screen:

- From the Top Line Menu select Edit, Clear Screen.

Clear Scrollback (Edit menu)

Clear data from the scrollback. Note, that if your data has not been saved to disk or sent to the printer, it will be lost.

To clear all data from the scrollback:

- From the Top Line Menu select Edit, Clear Scrollback.

Clear Both (Edit menu)

Clear data from the both the screen and the scrollback. Note, that if your data has not been saved to disk or sent to the printer it will be lost.

To clear all data from both the screen and the scrollback:

- From the Top Line Menu select Edit, Clear Both.

Copy Scrollback/Screen to Clipboard (Edit menu)

Copy scrollback or screen information to the Windows clipboard. In addition to copying information to the clipboard, it also remains on the screen or in the scrollback.

To copy scrollback or screen data to the clipboard:

Scrollback:

- From the Top Line Menu select Edit, Copy Scrollback to Clipboard.

Screen:

- From the Top Line Menu select Edit, Copy Screen to Clipboard.

Copy Scrollback/Screen to Disk File (Edit menu)

Copy scrollback or screen information to a disk file. In addition to copying information to the file, it also remains on the screen or in the scrollback.

To copy scrollback or screen data to a disk file:

Scrollback:
- From the Top Line Menu select, Edit Copy Scrollback to Disk File.

Screen:
- From the Top Line Menu select, Edit Copy Screen to Disk File.

When you copy screen data to a disk file, the Copy to Disk File dialog box displays.

Item	Purpose
File Name	Enter the name of a new file to which you want to copy the scrollback data. If you do not enter an extension, PowerLine assigns .cap.
List Files of Type	Display all files assigned the designated extension.
Directories	Select the appropriate directory.
Drives	Select the appropriate drive.

Copy Scrollback/Screen to Printer (Edit menu)

Copy scrollback or screen information to the printer. In addition to copying information to the printer, it also remains on the screen or in the scrollback.

To copy scrollback or screen data to the printer:

Scrollback:
- From the Top Line Menu select Edit, Copy Scrollback to Printer.

Screen:
- From the Top Line Menu select Edit, Copy Screen to Printer.

Copy Screen to Scrollback (Edit menu)

Copy screen information to the scrollback. In addition to copying information to the scrollback, it also remains on the screen.

To copy screen data to the scrollback:

- From the Top Line Menu select Edit, Copy Screen to Scrollback.

Preferences (Setup menu)

Activate or deactivate a variety of PowerLine program options.

To set PowerLine options:

- From the Top Line Menu, select Setup, Preferences. Respond to the prompts in the Preferences dialog box.

Preferences dialog box

Item	Purpose
System Status:	
Status Line	Activate Status Line display.
Tool Bar	Activate Tool Bar display.
Scroll Bars	Activate Scroll Bars display.
Keymap Bar	Activate Key Bar display.
Sound	Activate Sound capability.
Save window size on Exit	Save PowerLine's window on Exit.
Scrollback Size	Designate number of scrollback pages, from 5 to 150. Changing the scrollback size during a listing will clear the data in the current buffer.
Select Color	This option is dimmed if you activate ANSI color. Select the screen foreground and background colors by using the Select Color dialog box.

File Paths (Setup menu)

Define default paths where PowerLine should look for or store particular files. PowerLine displays these defaults when you perform particular operations such as creating a script. If you want a path different from the default, you can define it when performing a particular operation in the appropriate section of the dialog box.

To change default file paths:

- From the Top Line Menu select Setup, File Paths. Respond to the prompts in the File Paths dialog box.

File Paths dialog box

Item	Purpose
System	Path in which PowerLine will store hardware settings, listings, function key maps, and menus. You cannot change this path.
Capture	Path in which PowerLine will store captured files.
Script	Path in which PowerLine will store and search for script files.
Upload	Path in which PowerLine will search for files to upload.
Download	Path in which PowerLine will store downloaded files.

Protocol (Setup menu)

Set general protocol operations and specifics for Zmodem and ASCII transfers.
To setup protocol options:

- From the Top Line Menu select Setup, Protocol. Respond to the prompts in the Protocol Setup dialog box.

Protocol Setup dialog box

Item	Purpose
Receive Files Options:	
Overwrite	When receiving a file (using any protocol) overwrite a file that has the same name and path as the one you are downloading.
Rename existing file	When receiving a file (using any protocol) rename a file that has the same name and path as the one you are downloading by using the appropriate renaming convention.
Renaming Convention:	
Number [Test.Zip to Test1.Zip]	Rename the file by adding a number following the file name.
Letter [Test.Zip to TestA.Zip]	Rename the file by adding a letter following the file name.
Delete File on Abort	Delete a file when the transfer is aborted as a result of pressing the Cancel button during transfer or a communication problem.
ASCII Setup	See ASCII Protocol Setup.
Zmodem Setup	See Zmodem Protocol Setup.

Mouse (Setup menu)

Customize how your mouse works.

To set your mouse:

- From the Top Line Menu, select Setup, Mouse. Respond to the prompts in the Mouse Setup dialog box.

Mouse Setup dialog box

Item	Purpose
Double Left Click:	
Do Nothing	Do nothing in response to a double left mouse click.
Send Text With CR	Send all text on the line to the left of the mouse position with a carriage return in response to a double left mouse click.
Send Text Without CR	Send all text on the line to the left of the mouse position in response to a double left mouse click.
Send One Character	Send the character clicked on in response to a double left mouse click.
Right Click:	
Do Nothing	Do nothing in response to a right mouse click.
Position the Cursor	Position the cursor where you clicked in response to a right mouse click.
Send CR	Send a carriage return in response to a right mouse click.
Defaults	Resets mouse options to PowerLine defaults.

Dial Options (Setup menu)

Provide information PowerLine needs before dialing to make a connection.

To setup dialing options:

- From the Top Line Menu, select Setup, Dial Options. Respond to the prompts in the Dialing Setup dialog box.

Dialing Setup dialog box

Item	Purpose
Phone Number Prefix	If your phone system requires a number to access an outside line, enter it here. Also, if you need to cancel call waiting, you can enter the cancel code *70 here to deactivate call waiting while you are on line. Check with your local phone company to see if it supports *70.
Phone Number Suffix	Enter a phone number extension.
Local Area Code	Enter the local area code from which you will most often call. For listings, every number is entered as a long distance call. Before connecting, PowerLine checks to see your local area code and determines whether or not to dial 1 and/or the area code. By using this feature, you can easily change what is dialed by changing your local area code.
Number of Rings Before Answer	Specify the number of rings before the modem answers the phone.
Number of Redial Attempts	If the first attempt to dial is unsuccessful, PowerLine will redial. Specify the number of redial attempts. You can also set the number of redials during dialing from the Dialing Status dialog box.
Seconds Between Redial	Specify the number of seconds between redial attempts.

Port/Modem (Setup menu)

Change the communications port to which the modem is connected and select its appropriate settings. This information is set initially during installation.

To change the port and set it up:

- From the Top Line Menu, select Setup, Port/Modem. Select the appropriate port from the list of comm ports. Respond to the prompts in the Setup for COM Port dialog box.

Setup for COM Port dialog box

Item	Purpose
Type	Select the type of connection you will be using. Select Modem if you will be using a modem, Direct if you will be using a null modem cable or Int14 if you will be using a network pool. You will need additional software drivers for an Int14 connection. See your network documentation for more information.
Speaker Volume	Set the appropriate volume for the modem speaker.
Modem	Select the appropriate modem. This information is initially designated when you install PowerLine.

Chat Window (Connect menu)

While using PowerLine you'll probably have conversations with individuals through information services or bulletin boards. This conversation is referred to as "chatting". By activating PowerLine's Chat Window you can easily separate the information you type from the information you receive. The chat window splits the PowerLine window into two sections. It places the information you receive from the other system in the top (PowerLine Terminal) window and the information you type in the bottom window. These sections are sizable and moveable by using standard Microsoft Windows window controls.

To toggle the Chat Window on and off:

- From the Top Line Menu select Connect, Chat Window.

Hardware (Setup menu)

Designate hardware specifications; the current selections display in the Status Line. For appropriate specifications, consult information supplied by the BBS, information service or person to which you are connecting.

To set your hardware:

- From the Top Line Menu select Setup, Hardware. Respond to the prompts in the Hardware Settings dialog box. (The default hardware settings file is Default.hdw.)

Hardware Settings dialog box

Item	Purpose
Data Size	Data size indicates the number of "bits" in a computer word; the standard setting is eight. See your modem manual, or check with the service you are using for word size settings.
Parity	Parity is an error-checking procedure. See your modem manual for information specific to your modem and check with service you want to connect to. Parity options are: **None**: Disable error checking. **Even**: Comparison of bits sent by sending and receiving computers; if both numbers are not even numbers, an error message displays. **Odd**: Comparison of bits sent by sending and receiving computers; if both numbers are not odd numbers, an error message displays. **Mark**: Set the parity bit to 1. **Space**: Set the parity bit to 0.
Stop Bits	The number of stop bits indicates the end of a data segment. Enter the number appropriate to the modem and the system to which you are connecting. Designates the number of bits used by the sending computer to tell the receiving computer that a character has been transmitted.
Baud Rate	Designate the modem transmission speed.
Connection	Designate the port to which the modem is connected.
Additional Commands	Additional modem commands; appends automatically to modem initialization string.
Flow Control	Set the appropriate hardware and/or software flow control. Flow control allows the software or hardware to regulate the flow of data between computers to avoid data loss. Most information services use Xon/Xoff. Select from the following: **Software:** **None**: Do not use any software flow control. **XON/XOFF**: Signals sent by the receiving computer to prevent data loss. XOFF stops transmission temporarily; when ready to receive more data, an XON signal is sent. This setting is the PowerLine default.

ETX/ACK: Another available type of software flow control.

Hardware:

None: Do not use any hardware flow control.

RTS/CTS: Use request to send and request to clear.

DTR/DSR: Use data terminal ready, data terminal receive.

Save As Display Save Hardware Option dialog box. Since you will be using these settings with a listing, so you may want to assign a name that will be easy to associate with the listing. For example, our hardware settings for the CompuServe listing is CIS. Enter the name of a new file, or select a name from the existing list, in which the designated hardware settings will be saved. PowerLine assigns a .hdw file extension. PowerLine saves the file and asks if you want to make these Hardware Settings active.

Terminal (Setup menu)

Use the Terminal option to set options that determine how your system should behave under various conditions. Also, see How to Emulate a Terminal for more information.

To set terminal options:

- From the Top Line Menu select Setup, Terminal. Respond to the prompts in the Terminal Setup dialog box. (The default terminal settings file is Default.set.)

Terminal Setup dialog box

Item	Purpose
Emulation	Lets your PC simulate a mainframe terminal; select appropriate terminal type.
Fonts	Designate on-screen font.
Local Echo	Displays, on screen, every character as it is typed.
Incoming Echo	Display on the calling side every character sent.
CR/LF Pair (receive)	Activate to convert all received line feed/carriage returns.
CR/LF Pair (transmit)	Activate to convert all transmitted line feed/carriage returns.
Break Length	Enter a value in milliseconds to use as the signal length.
Backspace Key	Select **Acts as a Backspace** to have the Backspace key act normally. Select **Acts as a Delete** to have the Backspace key send the ASCII Delete character instead.
80 Column Screen	Designate screen width of 80 columns.
132 Column Screen	Designate screen width of 132 columns.
Save As	Display Save Terminal Option dialog box. Since you will be using these settings with a listing, so you may want to assign a name that will be easy to associate with the listing. For example, our terminal settings for the CompuServe listing is CIS. Enter the name of a new file, or select a name from the existing list, in which the designated terminal settings will be saved. PowerLine assigns a .set file extension. PowerLine saves the file and asks if you want to make these Terminal Settings active.

Character Map - Outgoing/Incoming (Setup menu)

Use a character map to ignore or replace specific characters or to convert their case during your communications session. You might use a character map if you communicate with a remote system that requires special characters not easily transmitted from a keyboard, or transmits characters that might be misinterpreted in the standard PC environment or transmits all characters in upper- or lowercase. You can have one map to filter incoming information and another to filter outgoing information.

Incoming/Outgoing

To create and use the Character Map to map transmitted or received characters:
- From the Top Line Menu, select Setup, Character Map. Select Outgoing to map transmitted characters or Incoming to map received characters. Respond to the prompts in the Character Map

dialog box.

Filter Map dialog box

Item	Purpose
Original	Original character as it is transmitted/ received before being filtered.
Converted	Character to which the original character is converted before transmission/reception.
Open	Edit an existing Character Map by using the File dialog box.
Display:	
Decimal	Displays characters in their decimal format.
Hex	Displays characters in their hexadecimal format.
ASCII	Displays characters in their ASCII format.
Change:	Change the original character to the converted character when the character map is active.
From:	The character transmitted/received when the original character is being sent/ received.
To :	Your new choice of characters to transmit/receive when original character is sent/received.
SaveAs	Display Save Character Map dialog box. Since you will be using these settings with a listing, so you may want to assign a name that will be easy to associate with the listing. For example, you might name the character map for the CompuServe listing on CIS. Enter the name of a new file, or select a name from the existing list, in which the designated character map will be saved. PowerLine assigns a .txf file extension. PowerLine saves the file and asks if you want to make this Character Map active.

Function Keys **(Setup menu)**

Map function keys to make frequently used scripts, series of keystrokes or external programs just a key press or mouse click away. The map allows you to define a function key as a series of keystrokes, external program, or string of characters. For example, if you connect to an information service often, you can define one key to operate as the series of keystrokes used to download files, and another key as the series of keystrokes used to sign off from the system. Also, you'll notice PowerLine's default function key map allows you access to our ASCII text editor PowerPad.

To map a function key:

- From the Top Line Menu, select Setup, Function Keys. Respond to the prompts in the Function Keys dialog box. (The default keyboard map is Default.kbm.)

Function Keys dialog box

Item	Purpose
Active	Check this box to activate the function. If the key is active, this box is checked.
Display	Enter the name that will display on the key at the bottom of the PowerLine screen.
Type	Enter the name of the program, script to run, or series of keystrokes to perform when this key is selected.
Text or File Name:	Select what type of information you are providing for the key, a script, text string (series of keystrokes) or external program.
New	Clear the Function Keys dialog box and start a new keyboard map.
Open	Open an existing function key map (.kbm) file. Use the file dialog box to select a map.
SaveAs	Display the Save Function Key dialog box. Since you will be using these settings with a listing, so you may want to assign a name that will be easy to associate with the listing. For example, our function key map we use with the CompuServe listing is CIS. Enter the name of a new file, or select a name from the existing list, in which the designated function key map settings will be saved. If you do not provide an extension, PowerLine assigns a .kbm file extension. PowerLine saves the file and asks if you want to make this Function Key Map active.

View Scratch Pad (Edit menu)

Use PowerLine's Scratch Pad to temporarily save short listings or pieces of information you may want to access later. For example, if your BBS sends you a list of files, you can cut this list from the screen or scrollback and place it on the Scratch Pad. Access this list later in your listing from the Receive or Send File dialog box to request the files, without typing each name. See Add to Scratch Pad for information on placing information on the Scratch Pad.

To access and manipulate the scratch pad:

- From the Top Line Menu select Edit, View Scratch Pad. Respond to the prompts in the Scratch Pad Items dialog box.

Scratch Pad Items dialog box

Item	Purpose
Send	Send the marked line of information on the Scratch Pad to the other computer system to which you are connected.
Open	Display the Open Scratch Pad File dialog box to open another scratch pad.
Save As	Display the Save Scratch Pad File dialog box. Since you can use different Scratch Pads with different listings, you may want to assign a name you will associate with the listing. For example, if you will be using this Scratch Pad with your CompuServe listing, you might name it CIS. Enter the name of a new file, or select a name from the existing list, in which the designated Scratch Pad settings will be saved. If you do not provide an extension, PowerLine assigns a .mkf file extension. PowerLine saves the file and automatically activates this Scratch Pad.
Delete	Delete marked information from the Scratch Pad.
Clear All	Clear all information from the Scratch Pad.

Phonebook (Connect menu)

Create and delete phonebooks, and listings within each phonebook, using the Phonebook Manager. See How to Establish a Connection for more information on using the Phonebook.

To access the phonebook:

- From the Top Line Menu select Connect, Phonebook. Respond to the prompts in the Phonebook dialog box.

Phonebook dialog box

Item	Purpose
Phonebooks	Displays names of phonebooks, including those installed during PowerLine installation.
Phonebook Listings	Listings associated with highlighted phonebook.
Dialing Queue	User-created list of listings to be dialed in order. See Creating a Dialing Queue.
Show	Display detail of terminal, baud and settings for the highlighted listing.
Dial	Dial the highlighted number or the numbers in the dialing queue. PowerLine displays the Dialing Status dialog box. See Dialing Status .
Open	Load highlighted phonebook listing.
Sort Listings By	Sort listings in alphanumeric, most used and last used format.
Add to Queue	Add highlighted phonebook listing to the dialing queue.
New	Create a new phonebook or listing using the New Item dialog box.
Edit	Modify highlighted listing information using the Edit Phonebook Listing dialog box.
Delete	Delete the highlighted item using the Delete Item dialog box. Additionally, you can delete a phonebook or listing by dragging the phonebook or listing name to the trash can icon. See Using the Trash Can.

Dialing Status

When you attempt to connect, PowerLine displays the Dialing Status dialog box:

Item	Purpose
Listing	Name of the phonebook listing being dialed.
Number	Phone number being dialed.
Total redials	Number of times PowerLine will redial the number.
Script	Login script that will run upon connection ("None" if no script is attached).
Time started	Time dialing started (determined from your system).
Redial	Redial the number.
Cancel	End dialing attempt and return to PowerLine.
+5 Redials	Add 5 more redials to the number you specified in the Dialing Setup dialog box.

The Dialing Status dialog box changes when you dial more than one number by using a queue. See Creating a Dialing Queue.

Listing Setup and Options

Customize every listing you create by modifying the current listing settings. The Listing Setup column displays the system categories available for customization: Hardware, Terminal, Toolbar, Menu, Script, Outgoing Filter, Incoming Filter, Function Key Map, and Scratch Pad. The Available Items column displays available options for the highlighted category.

The current setting for each category of the listing is highlighted in the Available Items column as you move down the list of categories in the Listing Setup column. You can replace a current setting with an available option from the Available Items column by following the steps outlined below.

To modify the Listing's Current Setup

1. Highlight the category in the Listing Setup column that you want to modify.
2. The Available Items column displays the available options; use the mouse to move the cursor to the Available Items column. The cursor displays as a hand. Click, and drag the desired option left to the category in the Listing Setup column you want to modify.

Creating a Dialing Queue

A dialing queue can be useful when you want to dial several different numbers in succession.

To create a dialing queue:
1. From the Phonebook Listings section of the Phonebook Manager dialog box, highlight the category in the current listing setting that you want to add to the queue. The hand assumes a block shape.
2. Drag the block to the dialing queue section; when you release the mouse button, the listing name displays.

You can also drag an entire phonebook to the dialing queue.

When dialing a queue, the Dialing Queue Status dialog box changes to reflect multiple items:

Item	Purpose
Dial timeout in	Time remaining before timeout.
Remove Listing	Ends the current dialing listing, deletes that listing from the queue and dials the next listing in the queue.
Skip Listing	Skips the current dialing listing, leaves that listing in the queue and dials the next listing in the queue.
Cancel Queue	Cancels the entire dialing queue and returns to the PowerLine Terminal Window.

Using the Trash Can

In addition to deleting a phonebook or listing using the delete button from the Phonebook Manager dialog box, you can drag the name of the phonebook or listing to the trash can.

To use the trash can:
1. Highlight the phonebook or listing you want to delete. Click and drag the mouse; the hand assumes a block shape.
2. Drag the block to the trash can.

To view the contents of the trash can:
- Click on the trash can icon to display the contents of the trash can,

To restore the contents of the trash can to your Phonebook Manager:
- While viewing the contents of the trash can, select Restore.

Manual Dial (Connect menu)

Use Manual Dial when you want to call another system quickly or infrequently without setting up options or phonebooks.

To access quick connect:

- From the Top Line Menu select Connect, Manual Dial. Respond to the prompts in the Manual Dial dialog box .

Manual Dial dialog box

Item	Purpose
Phone Number	Position the cursor in the blank space on the telephone receiver; either type the phone number using the keyboard, or position the mouse on the desired number and click the left mouse button. As you click on each number, the number displays in the blank space on the telephone receiver.
Dial	Dial designated number. The Dialing Status dialog box displays.
Cancel	Cancels the quick dial.
Delete	Edit the phone number, one digit at a time, using the delete key; you can also edit the phone number by using the DEL key on the keyboard.
Clear	Delete the entire phone number.

Dial Current Listing

To dial the telephone number in the current communications session without using the Phonebook or Manual Dial options:

- From the Top Line Menu, select Connect, Dial Current Listing.

Send a File (Connect menu)

Use PowerLine to send entire spreadsheets, database files, word processing documents, or even computer programs. See How to Transfer Files for additional information.

To send files to another computer system:

- From the Top Line Menu, select Connect, Send a File. Respond to the prompts in the Send File dialog box.

Send File dialog box

Item	Purpose
File Name	Enter the name of the file you want to send. If you are using Zmodem, Ymodem or Kermit, you can send multiple files by entering the appropriate wild card here and using the Wildcard Send button.
List Files of Type	Display all files as specified.
Directories	Select or enter the appropriate directory path.
Drives	Select or enter the appropriate drive.
Protocol	A protocol is a standard way of regulating data transmission between computers. Select the protocol you want to use to send your file.
Scratch Pad	The Scratch Pad you have associated with the current listing. Click on the scroll bar to show the Scratch Pad contents. Mark any line in the displayed Scratch Pad to display it in the File Name box.
Wildcard Send	Automatically send files that match wildcard specifications (displayed in the Files box) for multiple file transfers. Use with Kermit, Ymodem Batch and Zmodem only.
Iconize During Transfer	Display an icon during the file send to indicate the send is complete.

Receive a File (Connect menu)

Use PowerLine to receive files from other computer systems or information services. See How to Transfer Files for additional information.

To receive files:

- From the Top Line Menu, select Connect, Receive a File. Respond to the prompts in the Receive File dialog box.

Receive File dialog box

Item	Purpose
File Name	Enter the name of a new file which you will be receiving. If you are receiving files by using the Kermit, Ymodem batch, or Zmodem this box will display the appropriate wildcard after you select the protocol.
List Files of Type	Display all files as specified.
Directories	Select or enter the appropriate directory path.
Drives	Select or enter the appropriate drive.
Protocol	A protocol is a standard way of regulating data transmission between computers. Select the protocol you want to use to send your file.
Scratch Pad	The Scratch Pad associated with the current listing. Click on the scroll bar to display the Scratch Pad's contents. Mark any line to display the text in the File Name box.
Iconize During Transfer	Display an icon during the file receive to indicate when the receive is complete.

PowerLink (Connect menu)

PowerLink, PowerLine's own protocol, allows bi-directional file transfer with on-line chat, to send and receive files between computers that both use PowerLine. View, select and transfer files between your computer and another system -- even an unattended one. If you need your files from work, there is no need to dash to the office after hours; just use PowerLink to retrieve the files from your office PC to your home PC quickly and easily. See How to Use PowerLink for additional information.

Using PowerLink

PowerLink users are identified as local or remote users. You are always the local user. The remote user may be a friend or colleague with whom you share data, or an unattended PC with PowerLine running. If your unattended PC will be accessed by other users, use the PowerLink Setup dialog box to designate the conditions under which your files may be accessed.

To access Powerlink:

- From the Top Line Menu, Select Connect, PowerLink. Respond to the prompts in the PowerLink DIalog Box.

PowerLink dialog box

Item	Purpose
Local Files	The files in the current path. The current path displays in Send Path.
File Size	Size of file highlighted in Local Files.
Send List	Files queued for sending. Click twice on a file name to place in the Send List.
Transfer Status	Depicts the percentage of transfer completion (send) for each file.
Send Path	Lists local user's current path.
Del From Send List	Delete highlighted file from Send List.
Del from Receive List	Delete highlighted file from Receive List.
Setup	Set access restrictions for callers to an unattended (remote) PC. Select exit options upon completion of a transfer. See Setting Up PowerLink.
Exit	Exit PowerLink.
Send Message	Activate to position cursor in Outgoing message area. Type a message, press Enter when message area is full to send that line of text to remote user. Continue entering text. The message will display in the Incoming message area of the remote PC.
Send Command	Queries the remote PC for status and displays status in the Incoming message area.
Incoming	Displays incoming message from the remote user.
Outgoing	Displays message as you type. Press Enter to send message. See also Send Message item.
File Size	Size of the file highlighted in the Receive List.
Remote Files	List files available to remote user within a designated path.
Receive List	Lists files queued for receiving. Click twice on a file name to place in the Receive List.
Transfer Status	Depicts the percentage of transfer completion (receive) for each file.
Remote Path	Lists remote user's path.

Setting Up PowerLink

Use the PowerLink Setup dialog box to establish important access restrictions for callers to an unattended (remote) PC.

Item	Purpose
Allowed Paths	Identify paths to which remote users will have access. Refer to Browse item.
Wildcard Pattern	Allow the remote user access to all files of a specific type using a wildcard designation. For example, allow access to all bitmap files in the specified path, with a *.bmp designation.
Access Password	Enter the password, up to eight characters, a remote user must enter to gain access to PowerLink.
Require Password	Activate to require any remote user to enter the password before accessing files.
Allow ALL Paths	Activate to allow the remote user access to all paths. If this item is checked, it is not necessary to specify Allowed Paths and Wildcard Pattern items.
Exit Remote When Done	Activate to close PowerLink on the remote (unattended) PC, and return to PowerLine.
Exit To DOS When Done	Activate to close PowerLink on the local PC, and return to DOS.
Delete	Delete highlighted paths in Allowed Paths.
Browse	Designate specific paths to which remote user is allowed access. Click twice on desired directory to select it; click on OK to place path in Allowed Paths section.

Hangup Modem (Connect menu)

To disconnect from the current call:

- From the Top Line Menu, select Connect, Hangup Modem.

Initialize Modem (Connect menu)

To send the Modem initialization command defined in the Hardware Settings dialog box to the modem:

- From the Top Line Menu, select Connect, Initialize Modem.

Wait For Call (Connect menu)

Use PowerLine to answer incoming calls from another computer system. To do this, you'll have to set your computer to wait for a call.

To set your computer:

- From the Top Line Menu, select Connect, Wait for Call.

Send Break (Connect menu)

Break signals are often used to signal other computer systems.

To send a break signal whose length is defined in the Terminal Settings dialog box:

- From the Top Line Menu, select Connect, Send Break.

Local Echo (Connect menu)

Activate Local Echo, also known as duplex, to display each character as it is typed on screen.

To activate local echo:

- From the Top Line Menu, select Connect, Local Echo.

Run (Script menu)

Select Run when you want to run the commands in your compiled script file.

To run a compiled script:

- From the Top Line Menu select Script, Run. Respond to the prompts in the Run Script dialog box.

Run Script dialog box

Item	Purpose
File Name	Enter the name of the compiled script (.plc) you want to run.
List Files of Type	Display a list of files with the .plc extension or all the files from the designated path.
Directories	Select the appropriate directory.
Drives	Select the appropriate drive.
Run	Run the specified script.

Manager (Script menu)

The Script Manager is the doorway to the PowerLine Script environment. From here edit, compile, run or delete your scripts.

To access the Script Manager:

- From the Top Line Menu select Script, Manager. Respond to the prompts in the Script Manager dialog box.

Script Manager dialog box

Item	Purpose
File Name	Enter the name of the script source file (.pls) you want to open, compile, delete compile/run, or enter the name of the compiled script (.plc you want to run).
List Files of Type	Display a list of files with the .pls extension or any files from the specified path.
Directories	Select the appropriate directory.
Drives	Select the appropriate drive.
Open	Open the specified script source file.
Exit	Leave the dialog box.
Run	Run the specified compiled script.
Compile	Compile the script.
Compile/Run	Compile the specified script source file and run the resulting .plc script file.
Delete	Delete the specified script file.

Stop (Script menu)

Stops script activity.

To stop a running script or learn mode:

- From the Top Line Menu, select Script, Stop.

Learn (Script menu)

Use the learn feature to capture transmitted or received characters to a specified script file during a communications session.

To access learn mode:

- From the Top Line Menu, select Script, Learn. Respond to the prompts in the Learn Script dialog box.

Learn Script dialog box

Item	Purpose
File Name	Enter the name of a new script file to which you want to capture keystrokes and selections. Select an existing script file to append newly recorded keystrokes. If you do not specify an extension, PowerLine assigns .pls.
List Files of Type	Display a list of files with the .pls extension or any files from the designated path.
Directories	Select the appropriate directory.
Drives	Select the appropriate drive.

Menu Editor (Design menu)

Using the Menu Editor you can design your own Top Line Menus and associate them with a particular listing. For example, you may want to create a different Top Line Menu for your GEnie listing and another for your Enable listing.

To design a Top Line Menu:

- From the Top Line Menu, select Design, Menu. Respond to the prompts in the Design Menu dialog box.

Design Menu dialog box

Item	Purpose
Menu Text	Edit the text that displays for the menu options.
Function	Select what you want the menu option to do.
Filename	The name of the script file or external programs you want to run when the option is selected.
Next	Select the next menu option.
Insert	Insert a new menu option.
Delete	Delete the highlighted menu option.
Arrows	Adjust the menu option placement.
SaveAs	Display the Save Menu File dialog box. Since you can use different Top Line Menus with different listings, assign a name easy to associate with the listing. For example, our Top Line Menu associated with the CompuServe listing is CIS. Enter the name of a new file, or select a name from the existing list, in which the menu options will be saved. If you do not provide an extension, PowerLine assigns a .mnu file extension. PowerLine saves the file and asks if you want to make this Top Line Menu active.
Defaults	Display the PowerLine default menu options.

Toolbar (Design menu)

Design a new toolbar or edit the existing toolbar to fit your needs. Use different icons and change their functionality. PowerLine's drop and drag feature makes custom toolbars as easy as clicking the mouse. Since you can associate different toolbars with different listings, simplify your communications by assigning icons to frequently performed tasks for listings you use frequently.

To access the Toolbar option:
- From the Top Line Menu, select Design, Toolbar. Respond to the prompts in the Toolbar dialog box.

Related Topics:
Removing a Toolbar Icon
Adding an Icon to the Toolbar

Removing a Toolbar Icon

To remove a Toolbar icon:

- From the Toolbar Editor dialog box, drag the icon from the Current Toolbar to the Available Toolbar.

Adding an Icon to the Toolbar

To add an icon to the Toolbar:

1. From the Toolbar Editor dialog box, select the menu item you want to represent as an icon.
2. Drag and drop the icon you want to use from the Available Toolbar to the Current Toolbar.

Toolbar dialog box

Item	Purpose
Current Menu	Currently displayed Top Line Menu.
Current	Command performed when the icon is selected.
Menu Item	Menu Option associated with the icon.
Current Icon Bar	Current, active Toolbar.
Available	Name of available bitmap icons to use in your toolbar.
Bitmap path	Path and filename of icon bitmaps.
Available	Available icon bitmaps.
SaveAs	Display the Save Toolbar File dialog box. Since you can use different toolbars with different listings, so you may want to assign a name easily associated with the listing. For example, if you will be using the toolbar with your CompuServe listing, you might name it CIS. Enter the name of a new file, or select a name from the existing list, in which the menu options will be saved. If you do not provide an extension, PowerLine assigns a .tlb file extension. PowerLine saves the file and asks if you want to make this toolbar active.

Toolbar (Setup menu)

Select the properties you want your toolbar to have.

To configure the toolbar:
- From the Top Line Menu, select Setup, Toolbar. Respond to the prompts in the Toolbar Setup dialog box.

Related Topic:
Changing Default Toolbar Characteristics

Changing Default Toolbar Characteristics

During installation, you select which type of toolbar (graphic or text) you want to install. If at any time you want to change which type of default toolbar (default.tlb) you have, from DOS or by using the Windows File Manager copy either text.tlb or graphics.tlb to default.tlb.

Toolbar Setup dialog box

Item	Purpose
Current Toolbar	Currently displayed toolbar.
Button Status	Activate the selected icon in the toolbar.
Toolbar Button Size	Define the toolbar button size.
Toolbar Position	Position the toolbar at the top of the screen, or in any screen position you prefer.
Menu Item	Name of the menu item associated with the selected icon.
All On	Activate all icons.
All Off	Deactivate all icons.
Delete	Delete the selected icon.

Index (Help menu)

To access the on-line help index:

- Select Help, Index from the Top Line Menu
  -or-
  Press Alt+Z at the Terminal Window.

Scripts (Help menu)

To display help on PowerLine Basic script language commands:

- Select Help, Scripts from the Top Line Menu.
  -or-
  Select Help, Index from the Top Line Menu, then click on the highlighted "Scripts (PowerLine BASIC)" hypertext link.

About (Help menu)

The About menu option provides you with information pertaining to your copy of PowerLine including the serial number, release number, and other pertinent information.

To access the About dialog box:

- From the Top Line Menu, select Help, About.

Select Color dialog box

Item	Purpose
Foreground	Designate foreground color.
Background	Designate background color.
Sample Text	Displays your selections.

ASCII Setup dialog box

Set options to be used during an ASCII file transfer by using the ASCII Protocol Setup dialog box.

Item	Purpose
ASCII Transmit Settings	
Use as Line Pace Character	Enter the character you want PowerLine to use to determine a new line. Leave this option blank if you do not want a pace character.
Expand Tabs	Expand tabs to spaces.
Expand Blank Lines	Include more space for blank lines.
End of Line Options:	
Strip CR	Strip carriage return.
Add LF to CR	Add line feed to carriage return.
Ignore CR	Ignore carriage return.
Strip LF	Strip line feed.
Add CR to LF	Add carriage return to line feed.
Ignore LF	Ignore line feed.

Zmodem Setup dialog box

Set options to be used during a Zmodem file transfer by using the Zmodem Protocol Setup dialog box.

Item	Purpose
Transmit Options:	
Crash Recovery	If during a Zmodem transmission a problem results in a crash, activate crash recovery to have PowerLine resume transmitting the file where it left off before the crash occurred.
Original File Time Stamp	Activate if you want to maintain a received file's original time stamp. Otherwise, Zmodem will use the time stamp from your system when it places the file on your system.
Error Detection:	
16 Bit CRC	Set the error detection rate to 16 Bit CRC.

	This rate is slower, but more reliable.
32 Bit CRC	Set the error detection rate to 32 Bit CRC. This rate is faster, but less reliable.
Transmit Method:	
Streaming	Transmit data in a continuous stream.
2K Window	Transmit data in 2048 byte blocks.
4K Window	Transmit data in 4096 byte blocks.

Edit Phonebook Listing dialog box

Use the Edit button to modify the currently highlighted listing. When you select edit, the choices you made when defining a listing display in the dialog box. Modify any of the designations, or add a new entry to the phonebook.

Item	Purpose
Listing Name	Enter listing name -- up to thirty characters.
Phone Number	Enter the service phone number. Enter commas to pause dialing for two seconds each.
Login Name/ID	Enter your User ID. A User ID is not required.
Password	Enter your password; the password will display as asterisks.
Listing Setup	See Listing Setup and Options .
Available Items	See Listing Setup and Options .
New	Create new hardware settings for designated entry.
Edit	Edit existing hardware settings for designated entry.
Replace	Modify listing setup with an available item.
Delete	Delete highlighted setting option.
Protocol	Selects the current listing's default protocol for uploading and downloading a file.
Use Dialing Prefix	When checked, the dialing prefix is sent out when dialing the phone number. When not checked, the prefix isn't sent.
Data Call	Modem call.
Voice Call	Functions as an autodialer; displays information on screen about call status.

New Item dialog box

Creating a Phonebook

To create a phonebook:

1. From the Phonebook Manager dialog box, select New.
2. The New Item dialog box displays.

Item	Purpose
Enter the name of the phonebook, up to thirty characters.	Identify phonebook.
Select Phonebook.	Indicate that this is a phonebook entry.

Creating a Listing

To create a phonebook listing:
1. From the Phonebook Manager dialog box, select New.
2. The New Item dialog box displays.

Item	Purpose
Enter the name of the listing, up to thirty characters.	Identify listing.
Select Listing entry.	Indicate that this is a listing entry.
Delete	Delete highlighted setting option.

Delete Item dialog box

Select Phonebook to delete the entire phonebook, or Listing to delete the highlighted listing entry. Select OK to delete, or cancel if you change your mind.

PowerLine BASIC Script Language

Use PowerLine Scripts to automate repetitive tasks you perform frequently. A script is a compiled ASCII file composed of a series of easy to use commands stored that can be executed anytime by using a minimum number of keystrokes.

Scripts can echo procedures you normally perform from the keyboard, or perform complex tasks and programming procedures. For example, you may design a script that automatically uses a pre-designed listing to dial and connect with your favorite information service and automatically download particular files each day from your electronic mail box. Also, use scripts to integrate information between Microsoft Window's applications by using DDE, allow your PC to act as a Mini-BBS or host by using the Host script or customize the way in which you use PowerLine by setting system options.

PowerLine BASIC Procedures

Create your scripts by either having PowerLine "Learn" your actions, or by programming from scratch:

Use the PowerLine Log File to keep track of your connections:

Learning Scripts

Use the learn feature to record transmitted/received characters during a communication session. You can later use those characters to perform a specific task. While in learn mode each character as it is received or transmitted is automatically recorded in the script file you specified. Because you are performing the procedures that the script will automate, you can see on the screen exactly what will happen when the script runs. This feature also minimizes syntax errors because it automatically records the keystrokes you specified. For example, if your script will allow you to download a particular file frequently from your favorite information service, you execute the necessary steps to download the files while PowerLine records them.

To activate learn mode:

1. From the Top Line Menu, select Script, Learn. The Learn Mode Dialog Box displays.
2. From the Learn Mode Dialog Box, enter a file name for your script at the Filename Prompt. If you do not specify an extension, PowerLine assigns .pls.
3. Select OK. PowerLine will begin recording your keystrokes and selections to the file you specified. While Learn mode is active, `Learning to the script` *`filename`* displays in the Status Line and a check mark displays next to the Learn Menu option.
4. When you have completed recording your selections/keystrokes, deactivate learn mode to stop recording. From the Top Line Menu, select Script, Learn. PowerLine stops recording and saves your script.

Now that you have completed recording your script, you may want to edit it by using Program Mode. You'll then need to compile it and run it.

Programming Scripts

Use PowerLine's Script Manager to create new scripts and edit scripts you have previously recorded.

Elements of a Script

Some basic script elements include:

- Commands
- Functions
- Variables
- Operators

Create/Edit Script
Format Script
Save Script
Compile Script
Run Script
Compile and Run Script

Commands (definition)

Commands are the backbone of your PowerLine script. You'll use commands to tell PowerLine to perform an action or process instructions in a particular manner. In addition to its own commands, PowerLine derives the syntax for many of its commands from the Microsoft Visual Basic programming language. For example, process a set of instructions by using a For..Next loop or the Select Case command. See Commands for a complete list.

Functions (definition)

Take advantage of PowerLine's functions to facilitate the processing of your script. Use functions by themselves as a formula, combined with other functions or within a command. Some functions perform simple calculations, others replace complex formulas. For example, use Chr$(123) to return the character whose ASCII decimal value is 123. See Functions for a complete list.

Variables (definition)

PowerLine scripts support the following types of variables:

- *Numeric*. Any combination of alphanumeric characters or numeric constants. For example, you might define the numeric variable X = 2.
- *Hexadecimal*. Hexadecimal constants. Indicate hexadecimal constants by using &H. For example, `x = &H10`.
- *Octal*. Octal constants. Indicate octal constants by using &O. For example, `x =&O20`.
- *String*. Any text data. Text assigned to variables are called string constants and are always enclosed by quotation marks. For example, you might define the string variable `Name$ = "Costa"`.
- *System*. PowerLine provides its own predefined system variables that can return information you may need frequently. For example, `$Baudrate` is a system variable that represents the current baud rate setting.
- *Arrays*. Array variables allow you to refer to and manipulate a list of values. Note that only single dimensional arrays are supported. For example:

```
For x: 1 to 10
Getstring (1,999)
Password$[x] = $InputString
Next x
```

- *User Defined.* PowerLine supports 20 user definable variables for integers and literals. Once you set these variables, their values remain set throughout your PowerLine listing and across scripts.

Operators (definition)

PowerLine supports the following:

Mathematical operators

(In order of precedence from highest to lowest):

Operator	Description
^	Exponentiation
-	Negative
*, /	Multiplication, division
\	Integer division
MOD	Modulo arithmetic
+, -	Addition, subtraction

Relational operators

Compare values, expressions, etc., by specifying multiple conditions using the following:

Operator	Description
>	Greater than
<	Less than
=	Equal to
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal to

Logical operators

(order precedence from highest to lowest):

Operator	Description
Not	Logical negation
And	And
Or	Inclusive Or
Xor	Exclusive Or
Eqv	Logical equivalence
Imp	Implication (first operand false or second operand true)

Truth Table

Use the following with And, Eqv, Imp, Or or Xor:

	T,T	**T,F**	**F,T**	**F,F**	(expression1, expression2)
And	T	F	F	F	
Or	T	T	T	F	
Xor	F	T	T	T	
Eqv	T	F	F	T	
Imp	T	T	T	T	

Create/Edit Script

To create a new script without using learn mode or to edit an existing script:

1. From the Top Line Menu select Script, Manager. PowerLine displays the Script Manager Dialog Box.
2. Select or enter the name of the script you want to open.
3. Select Open. PowerLine displays the script in PowerPad, PowerLine's script editing facility. Edit or enter your commands according to the format outlined in PowerLine BASIC Script Language.

Format Script

Keep the following general format rules in mind when creating a PowerLine script:

- Enter commands in either upper or lower case or a combination of both. The language is not case sensitive. For example, use either *GOSUB* or *GoSub.*
- Comment your script by preceding comment text with a single quotation mark ('). For example, *'This section initializes the modem.*
- Labels are indicated by a colon (:). For example, *Modem:*.
- Enclose text strings within double quotation marks ". For example, *filename"*.
- Arrays are one dimensional, they do not need to be declared or dimensioned. They will size themselves as they are used. Enclose the array index within square brackets [ ]. For example, *Password$[x] = $InputString*.
- Constants can be integer, floating point or strings. They do not need declaration. For example, *$ModemDialTone$ = ATDT".*
- Variables are alphanumeric up to 40 characters long. For example, *DialPause = 60*.
- Commands allow you to pass parameters, change variables and do not return values. For example, *If a>1 then Goto Init*.
- Functions always return a value. For example, *x = abs(-1)* returns the value of 1.

Save Script

Once you have completed creating or editing your script, save it by selecting File, Save from the PowerPad Top Line Menu.

Compile Script

Now that your source script file is complete, you must compile it; allow it to be put in a form that PowerLine can understand and use. When PowerLine compiles a script, it saves the script under the same name in two formats; source and compiled. PowerLine uses the compiled format (.plc file) to perform the tasks in your script. The source file (.pls file) stores the script commands so you can edit them later. To compile a script:

1. From the Top Line Menu select Script, Script Manager. The Script Manager dialog box displays.
2. Enter or select the name of the script (.pls file) you want to compile at the File Name prompt.
3. Select the Compile button.

PowerLine assigns a .plc extension to the compiled script and leaves the source .pls script file for you to edit at any time. If the compile is successful, PowerLine signals. Select OK. PowerLine creates a compiled (.plc) script from your source (.pls) script file.

If PowerLine detects an error in any of your script commands, PowerLine signals by displaying an error box that describes the error along with the line number on which the error occurred. Select OK and edit the script source file (.pls) accordingly. Edit the script again by using PowerPad. Use the PowerPad Search, Go to line # menu option to search the line number that contains the error. Edit your source (.pls) script file. Recompile and run the script.

Remember, anytime you want to change a script you must edit its source file and recompile it.

Run Script

To run a compiled script:

1. From the Top Line Menu, select Script, Run. The Run Script dialog box displays.
2. Enter or select the name of the compiled script (.plc file) at the Run Script dialog box. PowerLine begins performing the commands in the script.

Compile and Run Script

If you are fairly certain your script will not encounter any errors during compile, you can compile and run it by using one step:

1. From the Top Line Menu, select Script, Manager.
2. Enter or select the name of the sourced script file (.pls file) at the Script Manager dialog box.
3. Select the Compile/Run button. PowerLine assigns a .plc extension to the compiled script. PowerLine will also use this name with the .PLS extension for the source file. If the compile is successful, PowerLine signals. Select OK. PowerLine creates a compiled script (.plc) file from your source (.pls) script file and runs the script. If any errors occur PowerLine displays an error box. Select OK and edit the source file. Use the PowerPad Search, Go to line # menu option to search the line number that contains the error. Edit your source script file (.pls). Recompile and run the script.

Activating the Log File

Activate the PowerLine system log to record when you connected, disconnected and what listing you used each time you connect. To activate the system log you can either edit the initialization file or have a PowerLine script do it for you.

To use the PowerLine Script:

1. From the Top Line Menu select Script, Run.
2. Select LogFile.plc. Follow the on-screen prompts. The script will prompt you for the file you want to use as a log, automatically update the initialization file and activate the log.

To edit the initialization file:

1. Edit the PowerLine.ini file by using PowerPad or your favorite ASCII text editor.
2. Enter the name of the log file you want to use under the heading LogFile.
3. Save the PowerLine.ini file.

Each time you make a call, the time, date and where you connected will be added to the log file.

PowerLine BASIC Language Summary

The following Language topics are available:

Conversion

Asc
Chr$
Str$
Val

Date/Time

Timer	Weekday
$Date	$Friday
$Monday	$Saturday
$Sunday	$Thursday
$Tuesday	$Wednesday

Display

DisplayUserText
Local
SetLocalEcho
SetTextPosn
SetUserFgColor
SetUserBgColor
SetWindowColor
$TextCol
$TextRow

Dynamic Data Exchange

DDEConnect
DDEDisconnect
DDEExecute
DDERequest
DDESend

Error Control

OnError

File Manipulation

ChDrive
Close
Dir$
FileToClip
Input
LOF
Name
Print
UserOpenFile
Write
$SysPath
ClipToFile
CurDir$
EOF
FreeFile
Kill
MkDir
Open
RmDir
WinDir$
$MarkListName

Graphics

Bitmap
ClearBitmap
EnableBitmap
MoveBitmap
UserBitmap

Math

Abs	Fix
Int	Random
Randomize	Redim
Sgn	Sqr

Mouse/ Keyboard/ Dialog Box

ClearHotSpot
ClearMouseClick
ClearPushButton
HotSpot
InputBox$
InputPasswordBox$
MsgBox
MovePushButton
PushButton
OnMouseClick
$LastKey
$HotSpot
$Mousex
$LeftButton
$RightButton
$Mousey
$WithinCommand

PowerLine Telecommunication

DropDTR
EnableKeyBar
EnableToolbar
LoadMenu
LoadListing
LoadToolbar
MenuCmd
MenuFunction
ModemResult
PauseLearnMode
ReadModemFile
ReceiveFile
RestoreLearnMode
Send
SendBreak
SendFile
SetBreakLength
SetDefProtocol
ShowTopMenu
StartLearnMode
StopLearnMode
$BaudRate
$BreakLength
$Carrier
$Comid
$DefaultProtocol
$HardwareName
$LearnMode
$MenuName
$ModemAreaCode
$ModemDisplay
$ModemLine
$ModemInitString1
$ModemInitString2
$ModemOperation
$ModemPrefix
$ModemRedial
$ModemRedialPause
$ModemSpeaker
$ModemSuffix
$ModemType
$ModemVolume
$Parity
$Password
$PhoneNumber
$ProtocolFinished
$ProtocolReturnCode
$RxFilterName
$StopBits
$TerminalName
$ToolbarName
$TxFilterName
$UserId

Script Flow

Case	Else
ElseIf	End
EndIf	EndSelect
EndWhile	ExitFor
ExitWhile	For
GoSub	GoTo
If	Next
Return	Select Case
Wend	While
MsPause	Pause
RunScript	WaitFor
WatchFor	$WaitFor

Sound

Beep
PlayWAV
StopWAV
$SoundDone

String Manipulation

EncryptString
GetString
Lcase$
Len
Mid$
Rtrim$
Space$
UCase$
InStr
Left$
Ltrim$
Right$
SendKeyString
String$
$InputString

Window

ClearUserWindow
ClearWatch
ClearWindow
CreateUserWindow
DeleteUserWindow
IconizeWindow
MoveUserWindow
MoveWindow
RestoreWindow
SetTitleBar
ShowStatusLine
ZoomWindow
$IsIconized
$IsZoomed
$TitleBar
$WinHeight
$WinWidth
$Winx
$Winy

Commands

PowerLine Script commands are the backbone of the script. Use them to tell PowerLine to process the script in a particular manner or to perform a particular task. PowerLine supports the following commands:

Beep	ExitFor	OnError
Case	ExitWhile	OpenDiskCapture
ChDir	For	ReadSystemIniFile
ChDrive	GoSub	Rem
Close	Go To	RmDir
CloseDiskCapture	If	Reset
Else	InputPasswordBox$	Return
ElseIf	Kill	Select Case
EncryptString	KillQueue	WEnd
End	LineInput	While
EndIf	MkDir	WriteIniFile
EndSelect	Name	WriteSystemIniFile
EndWhile	Next	

Functions

Functions are built in formulas you can use to facilitate your script. Use a function by itself as a formula, combine it with other functions or use it within a command. PowerLine supports the following functions:

Abs
Asc
Bitmap
Chr$
ClearBitmap
ClearHotspot
ClearMouseClick
ClearPushButton
ClearUserWindow
ClearWatch
ClearWindow
ClipToFile
CreateUserWindow
CurDir$
DDEConnect
DDEDisconnect
DDEExecute
DDERequest
DDESend
DeleteUserWindow
Dir$
Display
DisplayUserText
DropDTR
EnableBitmap
EnableKeyBar
EnableToolbar
EOF
FileToClip
Fix
FreeFile
GetString
HotSpot
IconizeWindow
Input
InputBox$
InStr
Int
Lcase$
Left$
Len
LoadMenu
LoadListing
LoadToolbar
Local
LOF
Ltrim$
MenuCmd
MenuFunction
Mid$
ModemResult
MoveBitmap
MovePushButton
MoveUserWindow
MoveWindow
Msgbox
MsPause
OnMouseClick
Open
Pause
PauseLearnMode
PlayWav
Print
PushButton
Random
Randomize
ReadIniFile
ReadModemFile
ReceiveFile
ReDim
RestoreLearnMode
RestoreWindow
Right$
Rtrim$
RunProgram
RunScript
Send
SendBreak
SendFile
SendKeyString
SetBaud
SetBreakLength
SetDefProtocol
SetEmulationName
SetLocalEcho
SetTextPosn
SetTitleBar
SetUserBgColor
SetUserFgColor
SetWindowColor
Sgn
ShowStatusLine
ShowTopMenu
Space$
Sqr
StartLearnMode
StopLearnMode
StopWAV
Str$
String$
Timer
Ucase$
UserBitmap
UserOpenFile
Val
WaitFor
WatchFor
WeekDay
WinDir$
Write
ZoomWindow

System Variables

In addition to the variables you can create and name yourself, PowerLine provides its own predefined system variables that can return information you may need frequently:

$BaudRate
$BreakLength
$Carrier
$ClientHeight
$ClientWidth
$ComId
CR
CRLF
$Date
$DefaultProtocol
$EmulationIndex
$EmulationName
$False
$Friday
$HardwareName
$HotSpot
$InputString
$IsIconized
$IsZoomed
$LastKey
$LearnMode
$LeftButton
LF
$MarkListName
$MenuName
$ModemAreaCode
$ModemDisplay
$ModemInitString1
$ModemInitString2
$ModemLine
$ModemOperation
$ModemPrefix
$ModemRedial
$ModemRedialPause
$ModemSpeaker
$ModemSuffix
$ModemType
$ModemVolume
$Monday
$MouseX
$MouseY
$Off
$On
$Parity
$Password
$PhoneNumber
$ProtocolFinished
$ProtocolReturnCode
$PushButton
$QueueCount
$RightButton
$RxFilterName
$Saturday
$ListingName
$SoundDone
$StopBits
$Sunday
$SysPath
$TerminalName
$TextCol
$TextRow
$Thursday
$Time
$TitleBar
$ToolbarName
$True
$Tuesday
$TxFilterName
$UserId
$WaitFor
$Wednesday
$WinHeight
$WinWidth
$WinX
$WinY
$WithinCommand

Linking Applications with Dynamic Data Exchange

In addition to using the Clipboard, exchange information between PowerLine and other Microsoft windows applications by using the Dynamic Data Exchange (DDE) capability in PowerBasic. Use DDE to create a link to another windows application and manipulate applications remotely. For example, download stock information from the Dow Jones Information Service and use DDE to automatically place it in your Excel spreadsheet.

Keep in mind that the application you want to communicate with must also support DDE and must be running in Microsoft Windows (minimized or in background) at the time you want to communicate. The way in which applications use DDE may differ slightly from the Microsoft standard, see the applications documentation for more information.

A DDE conversation contains the following elements:
- Client who initiates the conversation.
- Server who responds to the application.
- Topic, the topic of the conversation (usually a data file).

During a conversation, Microsoft Windows serves as the director, by directing messages to the appropriate recipient. An application can be involved in several simultaneous conversations, each is uniquely identified.

List of DDE tasks
Sample DDE script

DDE tasks:

DDE Task: Initiates a DDE conversation.
PowerLine: **DDEConnect**(*index*,*servername"*,*topic"*)
index is the user defined reference number from 1 to 10
servername is the name of server program to begin communications
topic is the name of topic to refer to in communications

DDE Task: Returns a string variable that contains data from the specified item in a DDE conversation.
PowerLine: **DDERequest**(*index*,*item"*)
index is the user defined number from 1 to 10
item is the name of item to get data from

DDE Task: Sends the specified command string during a DDE conversation.
PowerLine: **DDEExecute**(*index*,c*mdstring*")
index is the user defined reference number from 1 to 10
cmdstring are the commands to execute on the server. Syntax must follow DDE rules as defined by Microsoft Windows where; [command](parameters).

DDE Task: Closes or ends a DDE conversation specified by the index.
Syntax: **DDEDisconnect**(*index,...*)
index is the user defined reference number from 1 to 10

DDE Task: Places the specified data into the specified item during a DDE conversation.
PowerLine: **DDESend**(*index,item",data"*)
index is the user defined number from 1 to 10
item is the name of item to send data to
data is the actual data to place in item

Sample DDE script:

The following script uses DDE to send the phrase *PowerLine test* to column 1, row1 of an Excel spreadsheet:

```
                                  'Two connects are required
                                  '1st to the application
                                  '2nd to item in application
DDEConnect(1,EXCEL",SYSTEM")      'Connect to EXCEL by using
                                  'Microsoft Windows system connect
DDEConnect(2,EXCEL",SHEET1")      'Connect to the specific work
                                  'sheet in EXCEL
DDESend(2,R1C1",PowerLine Test")  'Place PowerLine statement in
                                  'cell R1C1 of SHEET1
A=DDERequest(2,R1C1")             'Request contents of the cell
MsgBox(A,0,DDE")                  'and display it in a message box
DDEExecute(1,%fsPower2{ENTER})    'Save the work sheet in EXCEL
                                  'as Power2
DDEDisconnect(2)                  'Disconnect in order connected
DDEDisconnect(1)
End
```

Abs

Purpose: Computes and returns the absolute value of the argument.
Syntax: **Abs**(*x*)
x = is a value
Comments: The argument must be numeric and the absolute value is always a positive number.
Example:
```
x = (a)-(b)
msgbox (str$(abs(x)))
```

Asc

Purpose: Returns the ASCII code of the first character in a string.
Syntax: **Asc** (*"str"*)
str is a text string
Comments: If the string value is null, a value of 0 is returned.
Example:

```
value = Asc ("EXIT")
If value = 69
msgbox ("E is the first character")
EndIf
```

Bitmap

Purpose: Specifies a graphic (bitmap file) to display. Parameters allow you to control where, when and how large the display should be. Loads the bitmap into memory.

Syntax: **Bitmap**(*index,"filename",x,y [,show,cx,cy]*)
index is a user defined number from 0 to 50 used to identify bitmap graphic.
filename is the name of .BMP file to display
x,y are the starting pixel coordinates to display at
show is the optional mode to initially display: 0 = Off, 1 = On; +32 display is relative to PowerLine window, +64 display is relative to Windows window.
cx,cy are the optional width and height for display

Comments: Bitmap is dependent on the monitor's resolution. Default is display upon execution. See EnableBitmap to turn bitmap display on and off.

Example:
```
Bitmap(1,"c:\demo\demo.bmp",5,10,1+32)
```

Chr$

Purpose: Returns a character from the ASCII code (decimal value).
Syntax: **Chr$**(*x*)
x is integer with a value between 0 and 255 inclusive
Comments: Use this function to send unreadable characters such as line feeds (decimal value of 10), carriage return (decimal value 13) or to store data in a string. It is the inverse of ASC.
Example:

```
Send (Chr$(13))
```

ClearBitmap

Purpose: Removes the correlating bitmap from display. If no indices are specified, all bitmaps are removed.
Syntax: **ClearBitmap**(*index,...*)
index is the optional list of indices for the bitmap graphic(s) to clear
Comments: Optional indices assigned by Bitmap. See Bitmap for more information.
Example: `ClearBitmap(1)`

ClearHotspot

Purpose: Clears the mouse click hotspot specified by the index. If indices are not specified, all mouse click hotspots are cleared.
Syntax: **ClearHotspot**(*index,...*)
index is the optional list of indices to clear.
Comments: Optional indices assigned by Hotspot. See Hotspot for more information.
Example: `ClearHotspot(1)`

ClearMouseClick

Purpose: Clears the mouse click specified by the index. If indices are not specified, all mouse clicks are cleared.

Syntax: **ClearMouseClick**(*index,...*)
index is the optional list of indices to clear as assigned by PushButton.

Example: `ClearMouseClick(1,10)`

ClearPushButton

Purpose: Deletes the PushButton specified by the index. If indices are not specified, all push buttons are cleared.
Syntax: **ClearPushButton**(*index,...*)
index is the optional list of indices to clear
Comments: Optional indices assigned by PushButton.
Example: `ClearPushButton(2)`

ClearUserWindow

Purpose: Clears the user window specified by the index. If no indices are specified, all windows are cleared.
Syntax: **ClearUserWindow**(*index,...*)
index is the optional list of indices to clear
Comments: Optional indices assigned by CreateUserWindow.
Example: `ClearUserWindow(1)`

ClearWatch

Purpose: Clears the watch function specified by the index. If no indices are specified, all windows are cleared.
Syntax: **ClearWatch**(*index,...)*
index is the optional list of indices to clear
Comments: Optional indices assigned by WatchFor.
Example: `ClearWatch(5,6)`

ClearWindow

Purpose: Clears the current window.
Syntax: **ClearWindow**()
Comments: Works much like the BASIC CLS command.
Example: `ClearWindow()`

ClipToFile

Purpose: Saves the clipboard contents to a specified file.
Syntax: **ClipToFile**(*"filename"*)
filename is the file to which clipboard contents should be saved
Comments: Returns 0 if file saved or returns not 0 if there is no text on clipboard, a bad file name or if the file name is unacceptable.
Example: `CliptoFile("Myclip.???")`

CreateUserWindow

Purpose: Creates a user window at the specified pixel coordinates sized to the specified pixel coordinates.
Syntax: **CreateUserWindow**(*index,x,y, [cx,cy]*)
Where: *index* is the reference number from 0 to 50 assigned to reference the window.
x,y are the upper left pixel coordinates to begin window display
cx,cy are the optional window width and height
Comments: Assigns the new window an index reference number, specifies at what pixel coordinates to display the window and sets its height and width. See ClearUserWindow and DisplayUserText for more information.
Example: `CreateUserWindow(1,50,100,50,70)`

CurDir$

Purpose: Returns the current default path for the specified drive. If no drive is specified, it returns the current path for the current default drive.

Syntax: **CurDir$***([drive])*

drive is the optional drive to get the current path for

Example:
```
File$=Cur(d:)
```
(Returns D:\PWRLINE\SCRIPT)

DDEConnect

Purpose: Initiates a DDE conversation.
Syntax: **DDEConnect**(*index*,*"servername"*,*"topic"*)
index is the user defined reference number from 1 to 10
servername is the name of server program to begin communications
topic is the name of topic to refer to in communications
Comments: Returns 0 if the connect was successful or less than 0 if an error was encountered.
Example:

```
DDEConnect (1,"Excel","SYSTEM")
```

DDEDisconnect

Purpose: Closes or ends the DDE conversation specified by the index.
Syntax: **DDEDisconnect**(*index*)
index is the user defined reference number from 1 to 10
Comments: Breaks the connection in a DDE link.
Example: `DDEDisconnect(1)`

DDEExecute

Purpose: Sends the specified command string during a DDE conversation.
Syntax: **DDEExecute**(*index*,*"cmdstring"*)
index is the user defined reference number from 1 to 10
cmdstring are the commands to execute on the server.
Comments: Executes a set of commands known to the program PowerLine is linked to.
Example: `DDEExecute(1,"%faPower2{ENTER}")`
(Executes File, SaveAs, Power2 and Enter.)

DDERequest

Purpose: Returns a string variable that contains data from the specified item in a DDE conversation.

Syntax: **DDERequest**(*index*,*"item"*)
index is the user defined number from 1 to 10
item is the name of item to get data from

Example: `DDERequest(2,"R1C1")`

DDESend

Purpose: Places the specified data into the specified item during a DDE conversation.
Syntax: **DDESend**(*index,"item","data"*)
index is the user defined number from 1 to 10
item is the name of item to send data to
data is the actual data to place in Item
Comments: Send information to a particular part or place in the program you are connected to.
Example: `DDESend(2,"R1C1","PowerLine Test")`

DeleteUserWindow

Purpose: Deletes the user window specified by the index. If no indices are specified all are deleted.
Syntax: **DeleteUserWindow**(*index, ...*)
index is the optional list of indices to be deleted
Comments: Optional indices assigned by CreateUserWindow function.
Example: `DeleteUserWindow(1)`

Dir$

Purpose: Displays file names that meet your specifications.
Syntax: **Dir$***([filespec])*
filespec is the optional string that contains the path and file specification such as * or ?
Comments: The first time you use Dir$ you must include a file specification. Subsequent calls without a file specification return the next file name. A null string, "", is returned when no more matching files are found.
Example:

```
File$=Dir$("Cis.hdw")
```

Display

Purpose: Turns the display on or off.
Syntax: **Display**($*mode*)
$mode is on or off. Also use System Variables $On or $Off. See System Variables.
Example: `Display($On)`

DisplayUserText

Purpose: Displays text at a specified location relative to the user window created.
Syntax: **DisplayUserText**(*index,col,row,"string"*)
index is the user defined reference number from 0 to 50
col is the starting pixel column for display
row is the starting pixel row for display
string is the text or variable with a string to display
Comments: See CreateUserWindow for more information.
Example:
```
DisplayUserText(1,10,15,"My message here")
```

DropDTR

Purpose: Tells the modem to drop the connection (carrier) with the other computer system by turning off the DTR signal.
Syntax: **DropDTR**()
Comments: Use this function only if your modem supports the DTR signal, and you have its switches set to recognize DTR. See your modem's documentation.
Example: `DropDTR()`

EOF

Purpose: Checks for the end of a file.
Syntax: **EOF**(*filenum*)
filenum is the number of the file you are checking
Comments: If it is the end of the file, returns $True , if not returns $False.
Example:
```
EOF(#1)
```

EnableBitmap

Purpose: Turns the bitmap display on and off.
Syntax: **EnableBitmap**(*index,on/off*)
index is the reference number (0-50) as assigned by Bitmap
on/off is the display indicator; 0 to disable, 1 to display bitmap. Also, you can use system variables $On or $Off. See System Variables.
Example: `EnableBitmap(1,1)`

EnableKeyBar

Purpose: Turns the function key keyboard map display on and off.
Syntax: **EnableKeyBar**(*state*)
state is 0 to turn the bar off or 1 to turn the bar on. Also, you can use system variables $On or $Off. See System Variables.
Example:
```
EnableKeyBar(1)
```

EnableToolbar

Purpose: Turns the Toolbar display on and off.
Syntax: **EnableToolbar**(*state*)
state is 0 to turn the bar off or 1 to turn the bar on. Also, you can use system variables $On or $Off. See System Variables.
Example:
```
EnableToolbar(1)
```

FileToClip

Purpose: Copy a file to the clipboard.
Syntax: **FileToClip**(*filename*)
filename is the name of file to copy to the clipboard
Example: `FileToClip("myfile.txt")`

Fix

Purpose: Truncates a decimal value to an integer value.
Syntax: **Fix**(*num*)
num is any decimal value
Comments: Negative decimal numbers will truncate towards zero.
Example: `Fix(-5.3)`
(Returns -5.)

FreeFile

Purpose: Returns the next available file number.
Syntax: **FreeFile**()
Comments: Use the FreeFile function with the Open command to omit the file number and avoid trying to open separate files with duplicate numbers.
Example:

```
NextFilename = FreeFile()
```

GetString

Purpose: Retrieves a string of a specified number of characters.
Syntax: **GetString**(*maxlen,waittime*)
maxlen is the maximum number of characters to retrieve up to 132
waittime is the number of seconds to pause
Comments: GetString will pause script processing until it encounters a carriage return, the maximum number of characters or until the wait time has elapsed. Retrieve the string by using the $InputString variable. See System Variables.
Example: `GetString (5,10)`
(Returns maximum of 5 characters into $Inputstring.)

HotSpot

Purpose: Defines the area where the mouse click will activate a command (relative to PowerLine's window).

Syntax: **HotSpot**(*index,label,x,y,cx,cy*)
index is the user defined reference number from 0 to 50
label is the subroutine label name to invoke when spot is clicked on
x,y are the pixel coordinates of upper left corner of hotspot
cx is the width of hotspot
cy is the height of hotspot

Comments: See ClearHotSpot for more information.

Example:
```
Hotspot(1,Exit,5,5,30,30)
```
(Places a hotspot at X5,Y5, 30 pixel square that branches to the Exit label if clicked on.)

IconizeWindow

Purpose: Iconizes the current PowerLine window.
Syntax: **IconizeWindow** ()
Comments: See also ZoomWindow.
Example:
```
IconizeWindow()
```

Input

Purpose: Returns information from an open file.
Syntax: **Input** *filenum, variable1, variable2, ...*
filenum is the file reference number of the file you want to retrieve information from
variable 1-... is the comma-separated list of items to get from the file
Comments: See also Open.
Example:
```
Input #1, Modem Index, Init$,Init2$
```

InputBox$

Purpose: Creates a dialog box that requires input from the user by either requesting text or a button selection.

Syntax: **InputBox$***("text" ["title" ["default"]])*
text is the text to display as static text in dialog box
title is the dialog box title bar text
default is the default user entry prompt text

Comments: Use this function when you want to display a dialog box prompt, wait for the user to perform an action (enter text or select a button) and return to the contents of the edit box. If you do not include a title, a default title will display in the title bar. If you do not include default, the edit box will be empty. If the user selects Ok or presses Enter, whatever is in the box is returned. If the user selects Cancel, an empty string is returned.

Example:
```
name = InputBox$("Please enter your name.","Name","Your name")
```

InStr

Purpose: Returns the position of the offset character of the comparison string with the searched string.

Syntax: **InStr**(*"string","cmpstring"*)
string is the string to be searched
cmpstring is the string to look for

Comments: The position number is counted from 1. If cmpstring is not within string 1, 0 is returned. The value returned by InStr depends on the value entered for string1 and cmpstring. Either can be string variables, string expressions or string literals.

Example:
```
i = InStr ($PhoneNum,$ModemAreaCode)
```

Int

Purpose: Truncates a decimal value to an integer value.
Syntax: **Int**(*num*)
num is any decimal number.
Comments: Negative decimal numbers will truncate away from 0.
Example: `Int (-5.3)`
(Returns -6.)

Lcase$

Purpose: Converts a string of characters to lowercase.
Syntax: **Lcase$**(*"string"*)
string is a text string
Comments: All characters, either upper of lower case are changed to lowercase.
Example: `Lcase$ ("PowerLine")`
(Returns powerline.)

Left$

Purpose: Returns the leftmost characters of the string for the specified length.
Syntax: **Left$**(*"string",len*)
string is the text string
len is the string length
Comments: If len is greater than the number of characters in the string, the entire string will be returned. If the value of len is 0, a null string (zero length) will be returned.
Example:
```
string = ("Testme")
newstring = Left$(string,4)
```
(Returns Test.)

Len

Purpose: Returns the number of characters in the string
Syntax: **Len**("*string"*)
string is a text string whose length you want
Comments: If the string contains blank spaces, it will be included in the count of the number of characters.
Example:
```
pnlen = Len(877-8600)
```
(Returns 8.)

LoadMenu

Purpose: Loads a new Top Line Menu (.mnu) and makes it active.
Syntax: **LoadMenu**("*filename"*)
filename is the new menu (.mnu) file
Example: `LoadMenu("Cis.mnu")`
(Loads the CompuServe menu options.)

LoadListing

Purpose: Loads a new listing and makes it active.
Syntax: **LoadListing**(*"filename"*)
filename is a .pbl file
Comments: This function is especially useful for dialing.
Example: `LoadListing("ESIBBS.PBL")`
(Loads the Enable BBS listing.)

LoadToolbar

Purpose: Loads a new toolbar and makes it active.
Syntax: **LoadToolbar**(*"filename"*)
filename is a Toolbar (.tlb) file
Example: `LoadToolbar ("Default.tlb")`
(Loads the default Toolbar.)

Local

Purpose: Displays a text string in your PowerLine window.
Syntax: **Local**("*string"*)
string is a text string
Example: `Local ("PowerLine")`
(Sends PowerLine to the terminal window.)

LOF

Purpose: Returns the size of the file in bytes.
Syntax: **LOF**(*filenum*)
filenum is the number of the file you are checking
Example:
```
size = LOF(#1)
```

Ltrim$

Purpose: Returns a string with leftmost spaces removed.
Syntax: **Ltrim$**(*"string"*)
string is a text string
Comments: See Rtrim$.
Example:
```
Lefttrim = Ltrim$(" PowerLine")
```
(Returns PowerLine.)

MenuCmd

Purpose: Executes a Top Line Menu option. Identify the option by using its position number.
Syntax: **MenuCmd**(t*opindex,subindex,...*)
topindex is the position of top line menu option indices begin at 0
subindex is the position of option in submenu including separator lines indices begin at 0
Comments: Use this function when you want to invoke a menu function to perform a task especially while writing a script to perform unattended tasks. Dividing lines count as positions.
Example: `MenuCmd(5,1)`

MenuFunction

Purpose: Executes a top line menu option. Identify the option by using its menu ID code.
Syntax: **MenuFunction**(*menucode*)
menucode is a PowerLine defined menu id as found in the .mnu file. See Appendix K: Menu Id Codes for a list of menu codes.
Example: `MenuFunction(100)`
(Executes Disk Capture.)

Mid$

Purpose: Parse information from a specified string.
Syntax: **Mid$**(*"string",start,len*)
string is the string or string variable
start is an integer or numeric value
len is an integer value or numeric variable that specifies the length of characters to be taken from string, it must be between 1 and the length of the string.
Comments: If the length is greater than the number of characters to the right of the start, all rightmost characters beginning with the position are returned. If the value of the start or length is zero or if the start is greater than the number of characters in the string, a null string is returned.
Example:

```
i$ = Mid$(PhoneNum$,i-1,1)
```

ModemResult

Purpose: Stores the integer value of the result code returned by the modem script.
Syntax: **ModemResult**(*code*)
code is the integer value to set the internal modem result variable
Comments: PowerLine looks for a code equal to or higher than 300 when a connection has been established or a code under 300 when there is no connection.
Example: `ModemResult(300)`

MoveBitmap

Purpose: Moves a specified bitmap to a new location.
Syntax: **MoveBitmap**(*index,x,y*)
index is the bitmap reference number (0-50) assigned by the Bitmap function.
x is the new X coordinate
y is the new Y coordinate
Comments: Use MoveBitmap to move stagnant bitmaps or to create an animation affect.
Example:

```
For x = 1 to 10
   newx = x + 10
   MoveBitmap (1 newx,10)
Nextx
```

MovePushButton

Purpose: Move a user defined button to a new location.
Syntax: **MovePushButton**(*index,x,y*)
index is the PushButton reference number as assigned by PushButton.
x is the new X coordinate
y is the new Y coordinate
Example: `MovePushButton (1,20,20)`

MoveUserWindow

Purpose: Moves a specified user window to a new location and optionally with a new size.
Syntax: **MoveUserWindow**(*index,x,y [,cx,cy]*)
index is the user window reference number as assigned by CreateUserWindow
x is the new X coordinate
y is the new Y coordinate
cx is the optional new width
cy is the optional new height
Example: `MoveUserWindow (1,20,20,200,200)`

MoveWindow

Purpose: Moves a PowerLine window to a new location and/or size.
Syntax: **MoveWindow**(x,*y [,cx,cy]*)
x is the new X coordinate
y is the new Y coordinate
cx is the optional new width
cy is the optional new height
Example: `MoveWindow (50,50,200,200)`

Msgbox

Purpose: Displays a message in a dialog box and returns a value based on which button a user selects.

Syntax: **Msgbox**(*"msg" [,type [,"title"]]*)

msg is the message you want to display in box up to 1024 characters or 255 straight characters with no spaces

type is the sum of the values (1 from each group described below) that describe the number and type of buttons to display, icon style and identification of default button. Select from the following values:

0 - OK
1 - OK/Cancel
2 - Abort/Retry/Ignore
3 - Yes/No/Cancel
4 - Yes/No
5 - Retry/Cancel
+ 16 - add Critical Icon
+ 32 - add Query Icon
+ 48 - add Warning Icon
+ 64 - add Info Icon
+ 256 - make 2nd button the default
+ 512 - make 3rd button the default

Use 1-5 for the number and types of buttons. Use 16-64 for the icon type. Use 256 and 512 to decide default button. If type is omitted a single OK button displays as a default with no icon.

title is the text for the title bar of the message box.

Comments: The following values are returned based on what the user selects:

1 -OK
2 -Cancel
3 -Abort
4 -Retry
5 -Ignore
6 -Yes
7 -No

Example:

```
Dothis = MsgBox("Want to quit?", 4+32,"To Quit or not to Quit")
If dothis = 6
   Go To Endscript
Endif
```

MsPause

Purpose: Pauses script processing for a specified number of milliseconds.
Syntax: **MsPause**(*milliseconds*)
milliseconds is the duration to pause where 1000 ms equals 1 second
Comments: MsPause(250)
Example:
```
Send (Beep)
   MsPause(100)
Send (Beep)
```

OnMouseClick

Purpose: Performs a particular subroutine when any mouse button is clicked.
Syntax: **OnMouseClick**(*label*)
label is the label name to go to when a mouse button is pressed
Comments: After the mouse click, PowerLine will branch to the specified subroutine.
Example: `OnMouseClick(Endscript)`

Open

Purpose: Opens an external file for input, output or to append information to.

Syntax: **Open** *filename*
[**For** *mode*]
[**Access** *access*]
[*lock*]
As *filenum*

filename name of the file you want to open

mode sets the mode you want to use for the open. Enter **Append** to append the information to the end of the specified file, **Input** to start sequential input mode or **Output** to start sequential output mode.

access sets the access type. Enter **Read** to open the file for read only, **Write** to enter the file for writing only or Read **Write** to open the file for both reading and writing.

lock restricts access by other processes. Enter **Shared** to allow any process on any machine to read and write, **Lock Read** to allow no other process read access to this file, **Lock Write** to allow no other process write access to this file.

filenum is an integer 1 - 255 that identifies this file. Use this number as an identifier when using other I/O functions.

Example:
```
Open "junk.dat" for input as #1.
```

Pause

Purpose: Pauses script processing for a specified number of seconds.
Syntax: **Pause**(*seconds*)
seconds are the duration in seconds to pause.
Example:

```
Send (Beep)
   Pause(2)
Send (Beep)
```

PauseLearnMode

Purpose: Pauses learn script mode if it is active.
Syntax: **PauseLearnMode**()
Example: `PauseLearnMode()`

PlayWav

Purpose: Plays a sound (.wav) file.
Syntax: **PlayWav**("*filename*")
filename is the name and/or path of the sound file you want to play. Include a path if the file is not in the PowerLine path.
Comments: To hear sound files you need a Microsoft Windows multimedia compliant device see Advanced Features, Getting Voice Help for more information.
Example: `PlayWav("c:\pwrline\sound\01.WAV")`

Print

Purpose: Writes to an open file in tab separate format.
Syntax: **Print** *filenum, variable1, variable2, variable3,...*
filenum is the file number you want to write to in tab format
variable 1-... is the comma-separated list of items to read from the file
Example: `Print #1, Myvar1, Myvar2, Myvar3`

PushButton

Purpose: Display a pushbutton in a specified location.
Syntax: **PushButton**(*index,"text",label,x,y [,cx,cy]*)
index is the user defined reference number from 0 to 50
text is the string to display in the button
label is the label name to go to when the button is pressed
x is the X coordinate
y is the Y coordinate
cx is the optional width
cy is the optional height
Comments: See MovePushButton and ClearPushButton.
Example:
```
PushButton (1,"Help",help,20,20,40,20)
```

Random

Purpose: Produces random numbers in a specified range. Returns a random number between 0 and the specified maximum number.

Syntax: **Random**(*maximum*)
maximum is an integer or numeric variable.

Comments: The maximum number is always positive. The same sequence of random numbers will be generated unless you start the random number generator each time by using the Randomize function.

Example:
```
Random(175)
```

Randomize

Purpose: Resets the random number generator.
Syntax: **Randomize**(*seednum*)
seednum is an integer
Comments: The same sequence of random numbers will be generated unless you start the random number generator each time by using Randomize. To change the sequence of random numbers generated each time the script runs, place Randomize at the beginning of the script.
Example:

```
Randomize(255)
```

ReadIniFile

Purpose: Retrieves information from PowerLine's initialization file, pwrline.ini.
Syntax: **ReadIniFile** (*inifilename, keyword, itemname*)
inifilename is the name of the initialization file including path if necessary
keyword is the name of the section you want to search
itemname is the name of the parameter whose information you want to read
Example: `A$ = ReadIniFile("Pwrline.ini","Editor","Name")`
Read the information following the Name= parameter in pwrline.ini.

ReadModemFile

Purpose: Accesses the modem data file (i.e. modem.dat) and sets the initialization string information for a specified modem by using the system variables $ModemInitString1 and $ModemInitString2.

Syntax: **ReadModemFile** (*"modem filename.ext"*,*type*)
modem filename.ext is the name of your modem information file (modem.dat)
type is the type of modem you are looking for

Comments: For more information on using the modem data file see Appendix A: The Modem Configuration Files. Use this command to alter the modem engine.

Example:
```
ReadModemFile("modem.dat",22)
```

ReceiveFile

Purpose: Receives a specified file by using a specified protocol.
Syntax: **ReceiveFile**(*"filename" ["protocol"]*)
filename is the name of the file to receive to
protocol is the optional protocol to use
Example: `ReceiveFile("Test.Zip","Zmodem")`

ReDim

Purpose: Redimension an array to a different size.
Syntax: **ReDim** *array name* [*newsize*],...
array name is the name of the array
newsize is the new array size, optionally include more than one array
Example: `ReDim Testarray [3],Testarray2 [5]`

RestoreLearnMode

Purpose: Reactivates script learn mode after a pause.
Syntax: **RestoreLearnMode**()
Example:

```
Endscript:
Display($ON)
RestoreLearnMode
End
```

RestoreWindow

Purpose: Restore iconized PowerLine to its standard sized window.
Syntax: **RestoreWindow**()
Example: `RestoreWindow()`

Right$

Purpose: Returns the rightmost characters of a string for the specified length.

Syntax: **Right$**(*"string",num*)
string is a text string
num is an integer or numeric variable that specifies the number of characters to be returned.

Comments: If *num* is greater or equal to the number of characters in the entire string, the entire string will be returned. If *num* is 0, the null string (zero length) will be returned.

Example:
```
Right$("PowerLine",5)
```
(Returns rLine.)

Rtrim$

Purpose:	Returns string with rightmost spaces removed
Syntax:	**Rtrim$**("*string*")
	string is a text string
Example:	`Rtrim$("PowerLine ")`
	(Returns PowerLine.)

RunProgram

Purpose: Runs an external program.
Syntax: **RunProgram**(*"filename"*)
filename is the name of the program (Microsoft Windows or DOS) to execute
Comments: PowerLine will leave the script to process the external program and return to the line following this line to begin processing script commands.
Example: `RunProgram("C:\DOS\EDIT.COM")`

RunScript

Purpose: Suspends the current script and branches to another script.
Syntax: **RunScript**(*"filename"*)
filename is the name of the compiled script(.plc) file to run
Example: `RunScript("c:\pwrline\getcom.plc")`

Send

Purpose: Sends a string to the modem.
Syntax: **Send**(*"string"*)
string is the text string you want to send to the modem
Example: `Send("ATZ")`

SendBreak

Purpose: Sends a break signal.
Syntax: **SendBreak**()
Comments: The length of the break signal is set by the SetBreakLength function.
Example:
```
SendBreak()
```

SendFile

Purpose: Sends a specified file by using the specified protocol.
Syntax: **SendFile**(*"filename" ["protocol"]*)
filename is the name of the file to transmit.
protocol is the optional protocol you want to use (ASCII, Xmodem, Xmodem CRC, Xmodem 1K, Ymodem Batch, Cserve B+, Kermit)
Comments: Use this function for file transfers.
Example: `SendFile("Test.zip", "Zmodem")`

SendKeyString

Purpose: Sends keyboard keys to PowerLine.
Syntax: **SendKeyString**(*"string"*)
string is the text string to send to PowerLine
Comments: The string may contain control characters following the hat symbol (^) or the following expressions characters enclosed by square brackets ([ ]):
[BS] = Backspace key
[TAB] = Tab key
[UP] = Up arrow key
[DOWN] = Down arrow key
[LEFT] = Left arrow key
[RIGHT] = Right arrow key
[PGUP] = Page up key
[PGDN] = Page down key
[ESC] = Escape key
[HOME] = Home key
[END] = End key
[INS] = Insert key
[DEL] = Delete key
[ENTER] = Enter key
[&] = Alt key
Example: `SendKeyString("[F10]","[Right]")`
`SendKeyString("[&F1]")` (Sends the Alt+F1 keystroke)
`SendKeyString("[&C]","[S]")` (Selects the Send a File option from the Connect menu)

SetBaud

Purpose: Sets the baud rate.
Syntax: **SetBaud** *baudrate*
baudrate is 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
Example:

```
If Connect_Speed = 9600
   SetBaud(9600)
Endif
```

SetBreakLength

Purpose: Sets the length of a break signal.

Syntax: **SetBreakLength**(*milliseconds*)
milliseconds are the number of milliseconds the breaklength is set for

Example: `SetBreakLength(800)`

SetDefProtocol

Purpose: Sets a default protocol for file receive and transmit.
Syntax: **SetDefProtocol**(*"protocol"*)
protocol is the name of the protocol to use as default (ASCII, Xmodem, Xmodem CRC, Xmodem 1K, Ymodem Batch, Cserve B, Cserve B+, Kermit)
Example: `SetDefProtocol("Zmodem")`

SetEmulationName

Purpose: Defines the current terminal emulation setting.
Syntax: **SetEmulationName** ("*emulation*")
emulation can be one of the following: TTY, VT52, VT100, VT220, VT320, ANSI.
Example:

```
If $EmulationName = "TTY"
   SetEmulationName ("ANSI")
Endif
```

SetLocalEcho

Purpose: Sets echo to screen on or off for the current listing. Set the indicator to 0 to turn echo off, 1 to turn echo on.

Syntax: **SetLocalEcho**(*indicator*)
indicator is 1 for on or 0 for off or use $On or $Off. See System Variables for more information.

Example:
```
SetLocalEcho(1)
```

SetTextPosn

Purpose: Sets the position for text at the specified location.
Syntax: **SetTextPosn**(*row,col*)
row is the text row value
col is the text column value
Example: `SetTextPosn(5,5)`

SetTitleBar

Purpose: Specify new text for a window title bar.
Syntax: **SetTitleBar**(*"string"*)
string is the replacement text for the PowerLine title bar
Example: `SetTitleBar("MyTitle")`

SetUserBgColor

Purpose: Set background color.
Syntax: **SetUserBgColor**(r*ed,green,blue*)
red is the amount of red color from 0 to 255
green is the amount of green color from 0 to 255
blue is the amount of blue color from 0 to 255
Comments: The default colors are 192,192,192 (grey).
Example: `SetUserBgColor(150,150,150)`

SetUserFgColor

Purpose: Set foreground color.
Syntax: **SetUserFgColor**(*red,green,blue*)
red is the amount of red color from 0 to 255
green is the amount of green color from 0 to 255
blue is the amount of blue color from 0 to 255
Comments: The default colors are 0,0,0 (black).
Example: `SetUserFgColor(130,170,130)`

SetWindowColor

Purpose: Sets the PowerLine terminal window color.
Syntax: **SetWindowColor**(*foreground, background*)
foreground is the color index from 0 to 15
background is the color index from 0 to 15
Example: `SetWindowColor(3,10)`

Sgn

Purpose: Returns whether the number is positive, negative or 0.
Syntax: **Sgn**(*num*)
num is a numeric expression
Comments: Returns 1 if num is positive, 0 if num is 0, and -1 if num is negative.
Example: `Sgn(10)`
(Returns 1.)

Sqr

Purpose: Returns the square root of a number.
Syntax: **Sqr**(*num*)
num is a numeric expression
Example: `Sqr(4)`
(Returns 2.)

ShowStatusLine

Purpose: Specify text to display on left side of the status line.
Syntax: **ShowStatusLine**(*"string"*)
string is the text to display on left side of status line
Example: `ShowStatusLine("Dialing " + PhoneNum$)`
(Returns Dialing 555-5555.)

ShowTopMenu

Purpose: To display special menus.
Syntax: **ShowTopMenu**(*index*[subindex])
index is the position from left to highlight
subindex is the optional position within the pull down menu
Comments: If the subindex is 0 or is not specified, a pull down menu will not display.
Example: `ShowTopMenu (3,2)`

Space$

Purpose: Returns a string of the specified number of spaces.
Syntax: **Space$**(*len*)
len is a numeric expression greater than 0
Example: `Space$(5)`

StartLearnMode

Purpose: Activates learn mode and stores keystrokes in the specified learn script (.pls).
Syntax: **StartLearnMode**(*"filename"*)
filename is the name of the script file to use for learn mode
Comments: You can also start learn mode by using the Learnoption from the Scripts menu.
Example: `StartLearnMode("c:\pwrline\scripts\learnit.pls")`

StopLearnMode

Purpose: Ends learn mode and closes the learn script.
Syntax: **StopLearnMode**()
Example: `StopLearnMode()`

StopWAV

Purpose: Ends playing the current WAV (sound) file
Syntax: **StopWAV**(*"filename"*)
filename is the name of the sound file
Comments: Once a WAV is stopped there is no resume capability.
Example: `StopWAV("c:\WAV\demo.wav")`

Str$

Purpose: Returns a string representation of the specified integer or floating point value.
Syntax: **Str$**(*num*)
num is a numeric expression
Comments: Allows you to change an integer or floating point value to a string.
Example:
```
numvar = 123
numstring = str$(numvar)
```
or you can say
```
numstring = str$(123)
```

String$

Purpose: Returns a string of a specified length filled with first characters from another string.
Syntax: **String$**(*len,"str"*)
len is a numeric expression that specifies the length of the returned string
str is the string whose first len character builds the returned string
Example:
```
sixstring = string$(6,"abcdefgh")
```
(Returns abcdef.)

Timer

Purpose: Returns the number of seconds since midnight by using the current system time.
Syntax: **Timer**()
Example:
```
Timenow = Timer()
```

Ucase$

Purpose: Converts a string of characters to uppercase.
Syntax: **Ucase$**(*"string"*)
string is a text string
Example: `allupper = $ucase("abcdefg")`
(Returns ABCDEFG.)

UserBitmap

Purpose: Displays a bitmap of optional size identified by a particular index at specified coordinates in a user created window only.

Syntax: **UserBitmap**(*index,"filename",x,y[,cx,cy])*
index is the user defined number reference from 0 to 50
filename is the name of bitmap file to display
x is the x coordinate in the window
y is the y coordinate in the window
cx is the optional width
cy is the optional height

Comments: See CreateUserWindow for more information.

Example:
```
UserBitmap (1 "Phone.Bmp",5,5,10,10)
```

UserOpenFile

Purpose: Uses the selected file in your script.
Syntax: **UserOpenFile**(*title*,*pattern1*,*pattern2 [,name]*)
title is the text to display in title bar
pattern1 is the text to display in files list box such as *.* for all files
pattern2 is the wildcard pattern to look for such as *.* for all files
name is the optional path and/or filename to display in the edit control (with or without wildcards).
Example: `File = UserOpenFile("Files","All files (*.*)","*.exe","C:\DOS")`

Val

Purpose: Converts a string to a number.
Syntax: **Val** (*"string"*)
string is the text string you want to convert
Comments: Converts string up to the first non-numeric character. The string may also include a minus sign or decimal separator.
Example:
```
StringtoNum = Val ("12.34ABCD")
```
(Returns the numeric value 12.34.)

WaitFor

Purpose: Sets the string and duration PowerLine line should wait for to be returned from the other computer system when $Waitfor is invoked.

Syntax: **WaitFor**(*"string",seconds*)
string is the text string you want to wait for
seconds is the number of seconds to pause

Comments: Will continue the script if the string seen in the Terminal Window matches the string defined. If the string is never found, the script will continue when the time elapses.

Example:
```
WaitFor("OK",60)
```

WatchFor

Purpose: Alerts PowerLine to watch for a particular string to be sent from the other computer system and if received, branch to a specified label.

Syntax: **WatchFor**(*index,"string",label*)
[**Return**]
index is the user defined reference number from 0 to 50
string is the text string to watch for.
label is the label name to branch to if string is found

Comments: Will branch from where ever it is in processing at the time the string is found and branch to the line immediately following the label. The Watchfor index is then cleared. If the label has a corresponding Return command, PowerLine will return to where it originally branched from when it encounters the Return. PowerLine will do a specified watch only once. See ClearWatch for more information.

Example:
```
WatchFor(1,"lookforme",foundit)
```

WeekDay

Purpose: Returns the day of the week as an integer from 1 - 7 for Sunday through Saturday.
Syntax: **WeekDay**(*day of week*)
day of week is the day Sunday through Saturday
Example:

```
If (WeekDay() =$Wednesday)
SendFile ("Wed.dat")
Endif
```

WinDir$

Purpose: Returns the current path for Microsoft Windows..
Syntax: **WinDir$**()
Example: `WinDir()`
(Returns C:\WINDOWS.)

Write

Purpose: Writes information to an open file in comma separated format.
Syntax: **Write** f*ilenum, variable1, variable2, variable3, ...*
filenum is the file reference number
variable 1-... is the comma-separated list of items you want to write
Example: `Write #1, Firstname,Lastname,Sex`

ZoomWindow

Purpose: Maximize PowerLine window.
Syntax: **ZoomWindow**()
Comments: Zooms the current PowerLine window to its maximum size. See IconizeWindow.
Example:
```
Msgbox ("Let's zoom the window")
ZoomWindow()
```

$BaudRate

Purpose: Represents the current baud rate setting.
Syntax: **$Baudrate**
Comments: Returns the baud rate at the time access.
Example:
```
currentbaud = $baudrate
```

$BreakLength

Purpose: Represents the current break length setting.
Syntax: **$Breaklength**
Comments: The break length can be set by the SetBreakLength function.
Example:
```
currentbreaklength = $BreakLength
```

$Carrier

Purpose: Represents the current carrier status.
Syntax: **$Carrier**
Comments: The carrier status is 0 for no carrier or 1 for carrier. Set by the DropDTR function.
Example:

```
If $carrier = 1
   GoSub Online
else
   GoSub NotOnLine
endif
```

$ClientHeight

Purpose: Represents the current height of the PowerLine client area.
Syntax: **$ClientHeight**
Comments: The height is measured in pixels relative to the PowerLine window.
Example:

```
Tall_enough:
   HowTall = $ClientHeight
   If HowTall < 400
      GoSub NotTallEnough
   else
      GoSub TallEnough
   endif
Return
```

$ClientWidth

Purpose: Represents the current width of the PowerLine client area.
Syntax: **$ClientWidth**
Comments: The width is measured in pixels relative to the PowerLine window.
Example:

```
Wide_enough:
   HowWide = $ClientWidth
   If HowWide < 350
      GoSub NotWideEnough
   else
      Gosub WideEnough
   endif
Return
```

$ComId

Purpose: Represents the current communications driver.
Syntax: **$ComId**
Comments: The ComId is an integer value 1-4 for the current communications port.
Example:

```
If $Comid = 1
   Gosub Comdriver1
EndIf
```

CR

Purpose: Represents the carriage return code - ASCII code 13. Carriage return advances the cursor to the beginning of the current line.
Syntax: **CR**
Example: `Send (CR)`

CRLF

Purpose: Represents the codes for carriage return and line feed - ASCII codes 13 and 10. Carriage return line feed advances the cursor one line to its beginning.
Syntax: **CRLF**
Example:
```
Send (CRLF)
```

$Date

Purpose: Sets the system date and returns the date.
Syntax: **$Date =** *"str"*
str = date to set system in the mm-dd-yy or mm-dd-yyyy form.
Comments: When you use this function to set the date, the date will remain in effect until you turn off the computer if you do not have a battery powered CMOS RAM that retains this information or change the date again. Be careful with this variable since it changes your system's date settings.
Example:

```
MsgBox ("Hello, today is" +$Date)
```

$DefaultProtocol

Purpose: Represents the current default protocol.
Syntax: **$DefaultProtocol**
Comments: Can be set by using the SetDefProtocol function.
Example:
```
ShowStatusLine ("Default Protocol is " + $DefaultProtocol")
```

$EmulationIndex

Purpose: Returns the current terminal emulation setting as an integer.
Syntax: **$EmulationIndex**
Comments: The following integers can be returned:
0 - TTY
1 - VT52
2 - VT100
3 - VT220
4 - VT320
5 - ANSI
Example:

```
If $EmulationIndex = 0
   MsgBox ("You are using TTY")
EndIf
```

$EmulationName

Purpose: Returns the current terminal emulation setting as a string.
Syntax: $EmulationName
Comments: Returns TTY, ANSI, VT100, etc. See Terminal for more information on emulation.
Example:

```
If $EmulationName = "ANSI"
   MsgBox ("You are using ANSI")
EndIf
```

$False

Purpose: Represents a value of 0
Syntax: **$False**
Example:

```
If x = $False
MsgBox ("x = 0 or false")
Endif
```

$Friday

Purpose: Represents the day of the week.
Syntax: **$Friday**
Example:

```
If (Weekday() = $Friday)
MsgBox ("TGIF!!")
```

$HardwareName

Purpose: Represents the currently loaded hardware file.
Syntax: **$HardwareName**
Example:
```
ShowStatusLine ("Hdware file is" + $HardwareName)
```

$HotSpot

Purpose: Represents the integer value of the hotspot the user clicked.
Syntax: **$HotSpot**
Comments: Returns the last index number of the last hotspot clicked on. See the HotSpot function for more information.
Example:

```
If $HotSpot = 1
Gosub get_name
endif
```

$IsIconized

Purpose: Represents whether the PowerLine window is iconized.
Syntax: **$IsIconized**
Comments: If the window is iconized, $IsIconized is 1, if it is not iconized, $IsIconized is 0.
Example:

```
If $isiconized = 1
   Msgbox (The window is iconized ...")
EndIf
```

$InputString

Purpose: Represents the string retrieved by using the Getstring function.
Syntax: **$InputString**
Comments: Holds the literal string when the GetString function has been used in the script.
Example:
```
Getstring (80,100)
ShowStatusLine($Inputstring)
```

$IsZoomed

Purpose: Represents whether or not the PowerLine window is zoomed.
Syntax: **$IsZoomed**
Comments: If the window is zoomed, $IsZoomed =1. If it is not zoomed $IsZoomed = 0.
Example:

```
If $IsZoomed = 1
   MsgBox ("The window is zoomed . . .")
Endif
```

$LastKey

Purpose: Represents the character of the last key press.
Syntax: **$LastKey**
Comments: Use this variable to retrieve keyboard input.
Example:

```
If $LastKey = y
   MsgBox ("You pressed y. . .")
Endif
```

$LearnMode

Purpose: Represents whether or not learn mode is active.
Syntax: **$Learnmode**
Comments: If learn mode is active, $LearnMode = 1. If it is not active, $LearnMode = 0.
Example:

```
If $Learnmode = 1
   ShowStatusLine ("Learnmode is active . . .")
Endif
```

$LeftButton

Purpose: Represents whether or not the left mouse button is clicked.
Syntax: **$LeftButton**
Comments: If the left button is clicked, $LeftButton = 1. If it not clicked, $LeftButton = 0.
Example:

```
If $LeftButton = 1
   ShowStatusLine (You clicked the left mouse button . . .")
Endif
```

LF

Purpose: Represents the code for Line Feed - ASCII code 10. Causes the cursor to advance to the next line.
Syntax: **LF**
Example: `(Send (LF))`
(Advances the cursor to the next line in the same column.)

$MarkListName

Purpose: Represents the name of the currently loaded Scratch Pad file.
Syntax: **$MarkListName**
Example:
```
MsgBox ("Current Scratch Pad file is" + $marklistname)
```

$MenuName

Purpose: Represents the name of the currently loaded menu file.
Syntax: **$MenuName**
Example:
```
MsgBox ("Your current menu file is" + $menuname)
```

$ModemAreaCode

Purpose: Represents the area code.
Syntax: **$ModemAreaCode**
Comments: Holds the value of the number for the area code entered in the Dialing Setup dialog box.
Example:
```
i = InStr($PhoneNum,$ModemAreaCode)
```
(Returns the offset where the $ModemAreaCode is in the string.)

$ModemDisplay

Purpose: Represents whether or not to display modem commands on screen.
Syntax: **$ModemDisplay**
Comments: If $ModemDisplay is 0 commands are not displayed, if $ModemDisplay is 1 commands will be displayed.
Example:

```
If $ModemDisplay = 0
   Display ($ON)
Endif
```

$ModemInitString1

Purpose: Contains the first initialization string from Modem.Dat file.
Syntax: **$ModemInitString1**
Comments: Use this variable with the ReadModemFile function. See Appendix A: Modem Configuration Files for more information.
Example:

```
ReadModemFile ("Modem.DAT",$ModemType)
Send $ModemInitString1
```

$ModemInitString2

Purpose: Contains the second initialization string from Modem.Dat file.
Syntax: **$ModemInitString2**
Comments: Use this variable with the ReadModemFile function. See Appendix A: Modem Configuration Files for more information.
Example:

```
ReadModemFile ("Modem.DAT",$ModemType)
Send $ModemInitString2
```

$ModemLine

Purpose: Represents the type of phone line as set in the Phonebook Entry dialog box.
Syntax: **$ModemLine**
Comments: If $ModemLine is 0 the line is pulse, if $ModemLine is 1 the line is tone.
Example:

```
If $ModemLine = 0
   num$ = DialPulse
Endif
```

$ModemOperation

Purpose: Represents the modem state.
Syntax: **$ModemOperation**
Comments: The following integers represent the modem state:
0 = Initialization
1 = Dial
2 = Hangup
3 = Autoanswer
Example:

```
Select Case $ModemOperation
   Case 0
      GoSub InitModem
   Case 1
      GoSub Dial
EndSelect
```

$ModemPrefix

Purpose: Represents the prefix digits before the phone number as set by the Dialing Setup dialog box.
Syntax: **$ModemPrefix**
Example:
```
num$ = $ModemPreFix + $PhoneNum + $ModemSuffix
```

$ModemRedial

Purpose: Represents the number of redial attempts if the first is unsuccessful as set by the Dialing Setup dialog box .

Syntax: **$ModemRedial**

Example:

```
For x = 1 to $ModemRedial
Send($Num)
Pause $ModemRedialPause
Next x
```

$ModemRedialPause

Purpose: Represents the number of seconds between redial attempts as set by the Dialing Setup dialog box.
Syntax: **$ModemRedialPause**
Example: `Pause($ModemRedialPause)`

$ModemSpeaker

Purpose: Represents whether the speaker is on or off as set by the Dialing Setup dialog box.
Syntax: **$ModemSpeaker**
Comments: If the speaker is on $ModemSpeaker is 1, if the speaker is off, $ModemSpeaker is 0.
Example:

```
If ($ModemSpeaker)
   INITSTRING3 = INISTRING3 + INISPEAKERON
Endif
```

$ModemSuffix

Purpose: Represents the modem's suffix as set by the Dialing Setup dialog box.
Syntax: **$ModemSuffix**
Example:
```
num$ = $ModemPreFix + $PhoneNum + $ModemSuffix
```

$ModemType

Purpose: Represents the modem id as in the MODEM.CFG file.
Syntax: **$ModemType**
Comments: Holds the value of the modem's index number from the Modem.dat file. If a direct connection is selected or you have not dialed, a 0 is returned. Use this variable to retrieve the initialization string from the modem file Modems.dat. See Appendix A: Modem Configuration Files for more information.
Example:

```
ReadModemFile ("modem.dat",$ModemType)
```

$ModemVolume

Purpose: Represents the modem speaker volume as set in the Dialing Setup dialog box.
Syntax: **$ModemVolume**
Comments: For a low volume $ModemVolume = 0, a medium volume $ModemVolume = 1 or high volume $ModemVolume = 2.
Example:
```
INITVOL = "L" + str$($ModemVolume)
```

$Monday

Purpose: Represents the day of the week as set by the system clock.
Syntax: **$Monday**
Comments: Use this variable when automating scripts. For example, you may want to call the Dow Jones information service on Monday.
Example:

```
If (WeekDay() = $Monday)
   GoSub Call_Dow
EndIf
```

$MouseX

Purpose: Represents the x coordinate where the mouse is located.
Syntax: **$MouseX**
Comments: This position is relative to the PowerLine window the mouse is in.
Example:

```
For x = 1 to 20
MoveBitmap(1,$mousex,y)
Pause(1)
Next x
```

$MouseY

Purpose: Represents the y coordinate where the mouse is located.
Syntax: **$MouseY**
Comments: This position is relative to the PowerLine window the mouse is in.
Example:

```
While $leftbutton =1
x = $mousex
y = $mousey
gosub showposition
WEnd
```

$Off

Purpose: Represents a value of 0.
Syntax: **$Off**
Example:

```
If ($ModemDisplay)
   Display ($On)
Else
   Display ($Off)
Endif
```

$On

Purpose: Represents a value of 1.
Syntax: **$On**
Example:

```
If ($ModemDisplay)
   Display ($On)
Else
   Display ($Off)
Endif
```

$Parity

Purpose: Represents the parity setting from the current listing.
Syntax: **$Parity**
Comments: Holds the value of the number for the parity entered in the Hardware Setup dialog box.
Example:

```
If $HardwareName = "CSERVE" and $Parity = "N"
GoSub Dial
EndIf
```

$Password

Purpose: Represents the password for the current listing.
Syntax: **$Password**
Example:
```
Send($Password)
```

$PhoneNumber

Purpose: Represents the phone number for the current listing.
Syntax: **$PhoneNumber**
Example:
```
ShowStatusLine("Redialing"+ $Phonenumber)
```

$ProtocolFinished

Purpose: Represents the current status for the protocol in use.
Syntax: **$ProtocolFinished**
Comments: If the protocol has finished, $ProtocolFinished = 1, if it has not finished $ProtocolFinished = 0.
Example:

```
If $ProtocolFinished = 0
ShowStatusLine ("Error Receiving File")
EndIf
```

$ProtocolReturnCode

Purpose: Represents a protocol return code that is used to track errors.
Syntax: **$ProtocolReturnCode**
Comments: If there are no errors, $ProtocolReturnCode = 0. If there are errors a negative number is returned.
Example:

```
If $ProtocolReturnCode < 0
ShowStatusLine ("The download has encountered an error . . .")
EndIf
```

$PushButton

Purpose: Represents the index number from the most recent push button that was clicked on. See the PushButton function for more information.

Syntax: **$PushButton**

Example:

```
If $Pushbutton = 1
   gosub pushbutton_subroutine
endif
```

$QueueCount

Purpose: Returns the number of listings in the dialing queue.
Syntax: **$QueueCount**
Comments: See the modem.pls script for an example.
Example:

```
If $QueueCount=0
   GoSub HangupModem
   GoTo EndScript
Else
   GoSub Dial
EndIf
```

$RightButton

Purpose: Represents whether or not the right mouse button was pressed.
Syntax: **$RightButton**
Comments: If the right button was pressed, $RightButton = 1. If the right button was not pressed $RightButton = 0.
Example:

```
If $rightbutton = 1
   ShowStatusLine ("You pushed the right button)
endif
```

$RxFilterName

Purpose: Represents the loaded receive filter file name.
Syntax: **$RxFilterName**
Example:
```
ShowStatusLine ("Receive Filter is "+ $rxfiltername)
```

$Saturday

Purpose: Represents the day of the week as set by the system clock.
Syntax: **$Saturday**
Comments: Use this variable when automating scripts. For example, call the CompuServe Information Service on Saturday.
Example:

```
If (Weekday() = $Saturday)
```

$ListingName

Purpose: Represents the currently loaded listing name.
Syntax: **$ListingName**
Comments: Use this variable to check if the correct listing is loaded.
Example:
```
Connection$ = "Connected to" Left$(listingname,8)
```

$SoundDone

Purpose: Represents whether or not the sound file has finished.
Syntax: **$SoundDone**
Comments: If the sound has finished, $SoundDone = 1, if the sound has not finished $SoundDone = 0.
Example:

```
wavfile = "d:\pwrline\sound\v" + num + ".wav"
playwav (wavfile)
While $sounddone = 0
WEnd
MsgBox ("WAV file is done")
```

$StopBits

Purpose: Represents the current stop bit settings as entered in the Hardware Settings dialog box.
Syntax: **$StopBits**
Example:
```
stopbits = str$($stopbits)
ShowStatusLine("stopbits = " + stopbits)
```

$Sunday

Purpose: Represents the day of the week as set by the system clock.
Syntax: **$Sunday**
Comments: Use this variable when automating scripts. For example, call the CompuServe Information Service on Sunday.
Example:
```
If (Weekday() = $Sunday)
```

$SysPath

Purpose: Represents the PowerLine System path.
Syntax: **$SysPath**
Comments: Use this variable to retrieve the PowerLine path in case it is on a different drive.
Example:
```
wavfile$ = $syspath + "sound\v10.wav"
```

$TerminalName

Purpose: Represents the currently loaded terminal file.
Syntax: **$TerminalName**
Example:
```
ShowStatusLine ("Terminal file is " + $terminalname)
```

$TextCol

Purpose: Represents the current text column.
Syntax: **$TextCol**
Example:

```
If $textcol = 10
   msgbox ("You are in the wrong column")
endif
```

$TextRow

Purpose: Represents the current text row.
Syntax: **$TextRow**
Example:

```
If $textrow = 10
   msgbox ("You are in the wrong row")
endif
```

$Thursday

Purpose: Represents the day of the week as set by the system clock.
Syntax: **$Thursday**
Comments: Use this variable when automating scripts. For example, call the GEnie Information Service on Thursday.
Example:
```
If (Weekday() = $Thursday)
```

$Time

Purpose: Represents the system time in the form HH:MM:SS at the time of assignment as a literal string and can reset the system time.

Syntax: **$Time**

Comments: Use this variable when automating scripts. Be careful with this variable, it can reset your system's time.

Example:

```
If $Time = 8:00:00
GoSub Start_Listing
EndIf
```

$TitleBar

Purpose: Represents the current title bar text.
Syntax: **$TitleBar**
Example:
```
ShowStatusLine ("The current title bar is "+ $Titlebar)
```

$ToolbarName

Purpose: Represents the currently loaded Toolbar file name.
Syntax: **$ToolbarName**
Example:
```
ShowStatusLine ("Current Toolbar "+ $ToolbarName)
```

$True

Purpose: Represents a value of 1.
Syntax **$True**
Example:

```
If x<$True
   Msgbox ("x = 1 or true")
EndIf
```

$Tuesday

Purpose: Represents the day of the week as set by the system clock.
Syntax: **$Tuesday**
Comments: Use this variable when automating scripts. For example, call Enable's BBS on Tuesday.
Example:
```
If (WeekDay = $Tuesday)
```

$TxFilterName

Purpose: Represents the currently loaded transmit filter file name.
Syntax: **$TxFilterName**
Example:
```
ShowStatusLine ("Transmit filter is" + $txfiltername)
```

$UserId

Purpose: Represents the user id as entered in the Phonebook Manager.
Syntax: **$UserId**
Example:

```
If $UserId ="  "
MsgBox ("Enter an Id in the PhoneBook Manager")
EndIf
```

$WaitFor

Purpose: Represents whether or not the WaitFor function was successful.
Syntax: **$WaitFor**
Comments: If the WaitFor was successful $WaitFor = 1, if it was not successful $WaitFor = 0.
Example:

```
WaitFor (OK",60)
If ($WaitFor)
   Result = 0
EndIf
```

$Wednesday

Purpose: Represents the day of the week as set by the system clock.
Syntax: **$Wednesday**
Comments: Use this variable when automating scripts. For example, call the Dow Jones information service on Wednesday.
Example:

```
If (Weekday() = $Wednesday)
```

$WinHeight

Purpose: Represents the height of the window in pixels.
Syntax: **$WinHeight**
Comments: Use this variable to find the center of the window before you display anything.
Example:
```
CenterH = $WinHeight/2
```

$WinWidth

Purpose: Represents the width of the window in pixels.
Syntax: **$WinWidth**
Comments: Use this variable to find the center of the window before you display anything.
Example:
```
Centerx = $WinWidth/2
```

$WinX

Purpose: Represents the x coordinate of PowerLine's window (within Microsoft Windows' window) in pixels.
Syntax: **$WinX**
Comments: Use this variable to retrieve the current window position.
Example:

```
windowlocationx = str$($winx)
windowlocationy = str$($winy)
msgbox ("x coordinate is +windowlocationx + ".")
msgbox ("y coordinate is +windowlocationy +".")
```

$WinY

Purpose: Represents the y coordinate of PowerLine's window (within Microsoft Windows' window) in pixels.
Syntax: **$WinY**
Comments: Use this variable to retrieve the current window position.
Example:

```
windowlocationx = str$($winx)
windowlocationy = str$($winy)
msgbox ("x coordinate is +windowlocationx + ".")
msgbox ("y coordinate is +windowlocationy +".")
```

$WithinCommand

Purpose: Represents whether or not the user clicked inside of a message box.
Syntax: **$WithinCommand**
Comments: If the user clicked while inside the message box, $WithinCommand = 1, if the user clicked outside the message box $WithinCommand = 0.
Example:

```
If $WithinCommand = 0
   Goto endscript
EndIf
```

Beep

Purpose: Produces a tone through the computer's speaker, usually to alert you to an event.
Syntax: **Beep**
Example:
```
If error_flag = 1
   Beep
Endif
```

Case

Purpose: Checks multiple conditions and executes subsequent commands if a condition is satisfied.

Syntax: **Case** *expression*
expression is the expression you want to test against the Select Case expression

Comments: If the expression in the Case statement matches the expression in the Select Case command, PowerLine executes the commands following the Case statement until it encounters another Case statement, Case Else statement or an End Select statement.

Example: See Select Case.

ChDir

Purpose: Changes to a new directory.
Syntax: **ChDir** *path/directory*
path/directory is the path/directory you want to change to
Example: `ChDir "C:\PWRLINE\SCRIPTS"`

ChDrive

Purpose: Changes the current drive.
Syntax: **ChDrive** *drive*
drive is the drive identifier
Example: `ChDrive "C:"`

Close

Purpose: Closes a file that is currently open. If no file is specified, closes all open files.
Syntax: **Close** *#filenum, #filenum, #filenum*
filenum is the file(s) you wish to close
Comments: See Open.
Example:
```
Close #1, #2, #3
```

CloseDiskCapture

Purpose: Stops capturing information and close the open disk capture file.
Syntax: **CloseDiskCapture()**
Example:

```
If $Carrier=0
   CloseDiskCapture()
Endif
```

Else

Purpose: Performs the commands below the Else command if the expression within the associated If command is false.
Syntax: **Else**
Comments: The commands below the Else command will only be evaluated if the expression within the If command is false.
Example: See If.

ElseIf

Purpose: Proposes another condition if the initial If statement is false.
Syntax: **ElseIf** *condition*
condition is an expression
Comments: The expression within the ElseIf command will only be evaluated if the expression within the If command is false. If the ElseIf expression is true, commands following the ElseIf command will be executed. If the expression is false, the program control will skip to the Else command, ElseIf command or to the line following the Endif command if no other Else statements are present.
Example: See If.

EncryptString

Purpose: Can be used to encrypt information, such as passwords.
Syntax: **Enc_Pass$=EncryptString(Pass$)**
Comments: For examples of encrypting passwords, a common security precaution, refer to the login scripts provided with PowerLine.

End

Purpose: Denotes the end of a PowerLine script.
Syntax: **End**
Comments: PowerLine automatically terminates the execution of subsequent script commands when it encounters End. PowerLine will ignore any commands following the End command when the script is compiled.
Example:
```
End_Script:
End
```

EndIf

Purpose: Marks the end of an If structure.
Syntax: **EndIf**
Comments: If the If structure is false and there are no Else statements, PowerLine will continue to process commands at the line following the Endif command.
Example: See If.

EndSelect

Purpose: Denotes the end of a Select Case statement.
Syntax: **EndSelect**
Comments: If none of the case statements match the Select Case statement, PowerLine will continue to process commands at the line following the EndCase command.
Example: See Select Case.

EndWhile

Purpose: Denotes the end of a While Command.
Syntax: **EndWhile**
Comments: When PowerLine encounters the EndWhile command, it will evaluate the While condition to see if it is still true. If it is true, commands after the While command will be processed. If the condition is false, PowerLine will continue to process commands that follow EndWhile. See also WEnd.
Example: See While.

ExitFor

Purpose: Exits a For..Next loop.
Syntax: **ExitFor**
Comments: Transfers control to the line following the Next command in a For..Next loop.
Example: See For.

ExitWhile

Purpose: Exits a While loop.
Syntax: **ExitWhile**
Comments: Transfers control to the statement following the WEnd or EndWhile command in a While block.
Example: See While.

For

Purpose: Performs a group of instructions a specified number of times.

Syntax: **For** *counter=start* **To** *End [Step Increment]*
instructions
[**ExitFor**]
Next *[counter]*
counter is the numeric variable loop counter
start is the initial counter value
end is the final counter value
step is the amount counter is changed each time the loop is completed. The increment defaults to 1 if it is not specified.
instructions are the statements to be executed

Comments: Use a For..Next loop when you have to set a number of iterations. The commands following the For command are executed until the Next command is encountered. The first amount specified in the For statement is added to and compared with the last amount. If the amount is still less than the end amount, the loop is repeated. Place an Exit For command within the For..Next loop to transfer control to the statement immediately following the Next statement. Also, you can also count down by specifying a negative step value.

Example:
```
For i = 1 to 10 step 1
   GetString (10,60)
If ($INPUSTRING) = "END"
   ExitFor
EndIf
Next i
```

GoSub

Purpose: Branches to a subroutine located at label name.
Syntax: **GoSub** *label*
[**Return**]
label is the name of the subroutine you want to run.
Comments: Tells PowerLine to perform the subroutine in the script identified by the label name that follows the GoSub command. PowerLine will process the commands grouped under this label name until it reaches a Return command. At the Return command, PowerLine will return to the command which immediately follows the GoSub command. Labels are indicated by a colon (:).
Example:

```
InitModem:
   If ($ModemSpeaker)
   INITSTRING3 = INITSTR3 + INISPEAKERON
   Endif
Return
Dial Modem:
   GoSub InitModem
   Return
```

Go To

Purpose: Branches to the specified label name.
Syntax: **Go To** *label*
label is the label name you want to branch to.
Comments: Labels are indicated by a colon (:). You cannot return from where the Go To statement was called.
Example:

```
If a = 10
Go To Endscript
Endif
```

If

Purpose: Establish a condition under which PowerLine should perform a particular action.

Syntax: **If** *condition instructions*
*[***Else** *instructions]*
If *condition*
[instructions]
[**ElseIf** *condition*
[instructions]]
[**Else**
[instructions]]
EndIf
condition is a test a variable or values
instructions are statements to be executed

Comments: Use the If command to perform simple tests on variables or values. If the expression is true, commands following the If command will be executed. If the expression is false, the program control will skip to the Else command, ElseIf command or to the line following the EndIf command if no Else statements are present.

Example:
```
number = 1
If number = 1
   msgbox ("Number equals One")
ElseIf number = 2
   msgbox ("Number equals Two")
Else
   msgbox ("Number isn't one or two")
EndIf
```

InputPasswordBox$

Purpose: Creates a dialog box that requires the user to enter (input)text or select a button.
Syntax: **InputPasswordBox$("text"[,"title"])**
text is static text that displays in the dialog box.
title is the dialog box's title bar text.
Comments: Use InputPasswordBox$ to obtain information from the user without displaying what the user types. This is particularly useful when requesting password information.

Kill

Purpose: Deletes specified files.
Syntax: **Kill** ([*path/specification*])
path/specification is the optional path and file specification for the files you want to delete
Example: `Kill "c:\news\article.txt"`

KillQueue

Purpose: Stops the dialing of a particular queued listing or all queued listings.

Syntax: **KillQueue(0)** stops dialing the listing currently in the queue and dials the next queued listing. **KillQueue(1)** stops dialing listings placed in queue.

Example:

```
If Cancel_Flag=0
   KillQueue(0)
Else
   KillQueue(1)
```

LineInput

Purpose: Reads one line at a time from an already open file and places that line into one variable. Use this command in conjunction with the Open command.

Syntax: **Open** "*filename"* **for Input as** *#1*
LineInput #1, *InputLine*

Comments: Only one variable can be used with LineInput.

Example:
```
Open "CSERVE.PBL" for Input as #1
   While not EOF (#1)
      LineInput #1, InputLine
   WEnd
```

MkDir

Purpose: Creates the specified directory.
Syntax: **MkDir** *path*
path is the path and directory identifier variable
Example: `MkDir "C:\NEWS"`

Name

Purpose: Renames or moves specified files.
Syntax: **Name**(*file name1*) **As** (*file name2*)
file name1 is the name, path or file specification for file(s) you wanted moved or renamed
file name2 is the name and/or path you want the file(s) changed to
Example:

```
Name "c:\news\myfile.txt" As "c:\news\yourfile.txt"
```

Next

Purpose: Tells PowerLine to increment and evaluate the associated For statement.
Syntax: **Next** *[counter]*
counter is the numeric variable loop counter
Example: See For.

OnError

Purpose: Capture run time errors.

Syntax: **OnError** *action*
action is the command you want PowerLine to perform when it encounters a runtime error.

Example:
```
OnError Goto My_Subroutine
```

Use the Goto 0 parameter to turn off the OnError command:

Syntax: **OnError Goto 0**

Example:
```
If Error_checking = "YES"
   OnError Goto 0
Endif
```

OpenDiskCapture

Purpose: Opens a disk capture file to store incoming data.
Syntax: **OpenDiskCapture** ("*filename*")
filename is the name of the disk capture file
Comments: This command is useful when creating a login script. It allows you capture the keystrokes you'll need for your script in a disk capture file.
Example:

```
If $Carrier=1
   OpenDiskCapture("Cserve.cap")
Endif
```

ReadSystemIniFile

Purpose: Reads the pwrline.ini file into memory. Use this command after altering pwrline.ini with the WriteIniFile command to allow changes to take effect.

Syntax: **ReadSystemIniFile()**

```
Example: WriteIniFile("Pwrline.ini","Editor","Name","Your name")
ReadSystemIniFile()
```

Rem

Purpose: Denotes your comments in a PowerLine Script.
Syntax: **Rem** *comment*
comment is the comment text
Comments: Rem allows you to enter any text comments within your script. Comments are useful to remind you how the script works and to provide you with information. They are ignored when the script is compiled and display exactly as you entered them when the source script file is listed. Also, you may use a single quotation mark instead of the Rem statement to indicate comments.
Example:

```
Rem Initialize the modem
```

RmDir

Purpose: Removes the specified directory.
Syntax: **RmDir** *path/directory*
path/directory is the path/directory you want to remove
Example: `RmDir "C:\NEWS"`

Reset

Purpose: Closes all currently open files.
Syntax: **Reset**
Example:
```
Reset
```

Return

Purpose: Indicates the end of a subroutine and returns to the GoSub or WatchFor function that called the subroutine.

Syntax: **Return**

Comments: When PowerLine encounters the Return used in combination with GoSub, it resumes processing the script commands immediately following the GoSub command that caused execution to branch to the specified label. When PowerLine encounters a Return command used in combination with a WatchFor function, it will return from the specified label and resume processing the commands immediately following the WatchFor function.

Example: See GoSub and WatchFor function.

Select Case

Purpose: Define the direction your script will take on an either or basis.

Syntax: **Select Case** *test expression*
Case *expression1 [To expression 16]*
[instructions]
[**Case** *expression2*
[instructions]]
[**Case Else**
[instructions]]
End Select
test expression is the expression compared with Case expressions
expressions are the expressions compared with test expression
instructions are the instructions performed when the expressions match

Comments: Depending on the value of the test expression, the script will execute specific instructions. Use the Select Case statement when you want to give your script more than one alternative route. When PowerLine encounters a Select Case command, it evaluates the expression. It then moves to the first Case statement, evaluates its expression. If that expression matches the Select Case expression, PowerLine executes the commands following the Case Statement until the next Case statement or until it encounters an End Select command. It then processes commands following End Select.
If the first Case expression does not match the expression in the Select Case Statement, PowerLine moves to the second and evaluates its expression. It continues until one of the expressions is true or until it encounters an optional Case Else statement. If a Case Else statement is included, its commands will be processed if none of the Case statements are true. If there is not a Case Else statement and none of the expressions following the Case statements are true, PowerLine begins processing the commands following End Select.

Example:
```
Select Case decision
   Case "Yes"
      msgbox("YES")
   Case "no"
      msgbox("NO")
   Case "maybe"
      msgbox("MAYBE")
Case Else
      msgbox ("No decision was made")
End Select
```

WEnd

Purpose: Denotes the end of a While structure.
Syntax: **WEnd**
Comments: When PowerLine encounters the Wend command, it will evaluate the While condition to see if it is still true. If it is true, commands after the While command will be processed. If the condition is false, PowerLine will skip to the command that follows WEnd. Interchangeable with EndWhile.
Example: See While.

While

Purpose: Executes statements in a block more than once.

Syntax: **While** *condition*
[*instructions*]
WEnd
condition is a test of variables or values
instructions are statements to be executed

Comments: The While commands denotes the beginning of a block of commands which will be processed as long as the condition is true. When PowerLine encounters a While command, it evaluates its expression. If the expression is true, it processes commands following While until it encounters a WEnd or EndWhile command. At the WEnd or EndWhile command, PowerLine will return to the While command and reevaluate its expression. If it is still true, the process is repeated. If it is not true, PowerLine will begin processing the commands immediately following the WEnd or EndWhile commands.

Example:
```
While count < 10
   count = count + 1
WEnd
```

WriteIniFile

Purpose: Writes information to PowerLine's initialization file.

Syntax: **WriteIniFile** (*inifilename, keyword, itemname, data*)
inifilename is the name of the initialization file including path if necessary
keyword is the name of the section you want to write to
itemname is the name of the parameter whose information you want to write to
data the information you want to write to file

Example: `WriteIniFile("Pwrline.ini","Editor","Name","Your Name")`
Write the information following the Name= parameter in pwrline.ini.

WriteSystemIniFile

Purpose: Writes any changes, made to PowerLine by using scripts, to the pwrline.ini file.
Syntax: **WriteSystemIniFile()**
Comments: Use this command when you want changes made by other commands to be added to the pwrline.ini file.
Example: WriteSystemIniFile

Glossary

ANSI
ASCII
Asynchronous communication
AT
Baud
BBS
Bit
Block
Break
Bps
Buffer
Button
Capture File
Carriage Return
Carrier signal
Character length
CCITT
Comm Port
Communications Settings
Control character
CRC
Data bits
Data Compression
Download
Echo
EOF
Error-checking
Filter Map
Flow control
Full-duplex
Half-duplex
Hayes Command Set
Handshaking
Host
Kermit
Line Feed
Local Echo
Log Off
Log On
MNP
Modem
Noise
Null modem cable
Packet
Parity
Protocols
Remote
Retries
RS-232C
Serial Interface (card)
Start/Stop bits
Sysop
Terminal Emulation
Terminal Settings
Upload
V.21
V.22
V.22bis
V.32
V.32*bis*
V.42
V.42*bis*
X-ON/OFF
Xmodem
Ymodem
Zmodem

ANSI
Signal that carries information using a continuously varying voltage, frequency or amplitude.

ASCII
Acronym for American Standard Code for Information Interchange. Used in virtually all personal computer data communications, the standard ASCII code set consists of 128 numbers ranging from 0 to 127, each of which has been assigned a particular meaning.

Asynchronous communication
Data communication of the start-stop variety. Each character is transmitted as a discrete unit with its own start bit and one or more stop bits.

AT
Prefix for many commands in the standard modem command set. The standard set is often called the Hayes AT command set.

Baud
A unit for measuring the speed of data transmission. Technically baud rates refer to the number of times the communications line changes states each second. Strictly speaking, baud and bits per second are not identical measurements, but many people use the terms interchangeably.

BBS
Bulletin Board System. An electronic system that allows users to call another computer in order to send/receive electronic mail and exchange files and information.

Bit
Acronym for "binary digit." The smallest unit of information in the computer world. Eight bits together are called a "byte," and four bits are called a "nibble."

Block
Stream of bits with a particular format that consist of data and control elements that is transmitted as a single unit. File transfer protocols divide a file into a series of blocks. Also called a packet.

Break
A signal sent to a remote computer system used to interrupt communication or to cause an action to be performed.

Bps.
Bits per second. The number of bits that can be transmitted in a second.

Buffer
A temporary "holding tank" inside your machine. A buffer actually consists of a number of memory chips that have been designated as such (dedicated to buffering task) by your communications software. In communications, buffers are most often used to capture incoming data.

Button
A spot on the screen that when selected performs an action or sets an option. Select the button by using the keyboard or the mouse.

Capture File
A file used to place incoming text to the disk so you can use it later.

Carriage Return
A control character that tells the computer to move the cursor to the beginning of the line. Send it by pressing Enter. Usually you must send a carriage return to the host computer before it will process a line.

Carrier signal
A signal whose characteristics can be altered to carry data. This is the signal a modem looks for before it will begin to send data.

Character length
A communications setting referring to the number of data bits in each character transmitted. Since seven data bits are required to transmit each character in the standard ASCII code set, the character length under these circumstances is seven. To transmit characters represented by numbers above 127, eight binary bits are required. The character length is eight.

CCITT
Consultative Committee for International Telephone & Telegraph.

Comm Port
Communications Port. The physical component in the computer through which it sends and receives data.

Communications Settings
Options that specify the parameters that allow the local and host computer to communicate, such as parity and stopbits.

Control character
A non-printing character designed to send a special signal (like a carriage return or linefeed) to a remote device. Control characters are generated by holding down the key, or your machine's equivalent, and pressing one of the letter keys. Like the other characters in the ASCII code set, control characters have standard meanings in most systems.

CRC
Cyclic Redundancy Check. A calculation is performed on a data packet before it is transmitted. After the packet is transmitted, the CRC "checksum" is sent, and the receiving computer compares the two. If there is a discrepancy, the data is re-transmitted.

Data bits
Also known as "information bits." These are the seven or eight bits that signify a character in asynchronous communications. The data bits of each character are always "framed" by a start bit, a parity bit (usually), and one or two stop bits.

Data Compression

Many modems offer data compression as a standard feature. The two most common types of data compression are MNP-5 and V.42*bis*. Each achieves compression by detecting repetitive information in the data stream and replacing it with "shorthand" information. The "shorthand" information is decoded by the receiving modem and translated back into the original data. This can speed transfer of easily-compressed data to, in the case of V.42*bis*, four times the original transmission speed. As a rule of thumb, ASCII text offers the highest rate of compression, while binary and already-compressed files offer little or no compression.

Download
To capture the information sent to your computer by another computer, as opposed to letting it disappear as it scrolls off the screen.

Echo
A mode of communication between computers in which the remote sends back to the local terminal every character typed on the local keyboard. Most computers usually echo characters when receiving text.

EOF
End of file. Occurs when the last piece of data from a file has been read.

Error-checking
A method used to verify the accuracy of data transmission.

Filter Map
A table used to translate particular received or transmitted characters.

Flow control
Regulate the flow of information to avoid data loss. Also called hand-shaking.

Full-duplex
A communications setting that enables you to send and receive information simultaneously the way you do when talking on the telephone.

Half-duplex
A communications setting in which data can travel in only one direction at a time. A special signal, equivalent to saying "over" when using a CB radio, must be sent at the end of each transmission to tell the receiving machine that it is free to transmit.

Hayes Command Set
Commands used to control the Hayes or Hayes compatible modem. Also called the AT Command Set.

Handshaking
The short ritual of exchanged signals two computers go through before communications can begin.

Host
A computer set up to receive calls from other computers and interact in a regulated environment.

Kermit
A file transfer protocol that features error correction, batch transfers and special text transfer capability.

Line Feed
A control character that advances the cursor or carriage to the next line on a printer of computer display.

Local Echo
A setting that causes every character typed both displays on the screen and is transmitted.

Log Off
The process of disconnecting from a remote system.

Log On
The process of establishing a connection and accessing a remote system.

MNP

Microcom Networking Protocol. A set of error control and data compression protocols. The most common MNP levels are:

2-4 Error control. Level 4 is the most common.

5 Data compression, with a theoretical limit of 2:1 compression.

Modem
An acronym for "modulator/demodulator." This is the device that translates the signals coming from your computer into a form that can be transmitted over standard telephone lines. A modem also translates incoming signals into a form that your computer can understand. Two modems, one for each computer, are needed for any data communications over telephone lines.

Noise
Electric pulses that may occur in telephone lines and interfere with communication.

Null modem cable
A cable designed to connect two computers via serial ports that are ordinarily connected to a modem. A null modem cable "fools" each system into thinking that it is actually talking to a modem instead of another computer.

Packet
Stream of bits with a particular format that consist of data and control elements that is transmitted as a single unit. File transfer protocols divide a file into a series of packets. Also called a block.

Parity
A form of error checking used to increase the 1s and 0s of the seven bits used to transmit a character and then adds an eighth bit. If the sum of the bits in the character is even, the system will add a 1 bit to make it odd. If the sum is odd to begin with, the system will add a 0 bit to leave it unchanged. Even parity works the same way. When no parity is used to transmit characters in the standard ASCII code set, the eighth bit is still transmitted, but it is ignored by both systems.

Protocols
A set of procedures that defines common data structures and conventions for sending and receiving for computer communication.

Remote
Computer that is not physically present but connected by using a modem.

Retries
Number of times a particular action has been tried unsuccessfully.

RS-232C
A standard developed by The Electronics Industry Association (EIA) specifying what signals and voltages will be used to transmit data from a computer to a modem. The full standard covers some 25 pins on the RS-232-C plug interface found on a serial card, but most personal computers make use of only a handful of these. The "C" is often dropped when using this term.

Serial interface (card)
A circuit board installed in a computer or word processor designed to convert the machine's internal parallel (eight-bits-at-a-time) communications into serial (one-bit-at-a-time) communications. The card includes an RS-232-C interface plug to accept the cable that connects your machine with your modem.

Start/Stop bits
In asynchronous communications, a start bit is transmitted at the beginning of each character to notify the receiving system that the next seven or eight bits will contain information. One or two stop bits are transmitted at the end of each character to tell the receiving system that the whole character has been sent and to prepare it for the next start bit. These bits are often called framing bits because they frame the data bits carrying the information.

Sysop
The system operator. The individual who operates and maintains a computer bulletin board system.

Terminal Emulation
The ability of a computer to emulate the characteristics of a specific terminal.

Terminal Settings
Settings to be used when emulating a terminal such as what type of terminal, font size, screen width, column width, etc.

Upload
To send information over the telephone lines to another computer directly from your floppy disk or cassette tape player, as opposed to typing at your keyboard.

V.21
CCITT-defined international standard for 300 bps data transfer over voice telephone lines. This differs from the Bell 103 standard used for 300 bps data communication in the United States.

V.22
CCITT-defined international standard for 1200 bps data transfer over voice telephone lines. This differs from the Bell 212A standard used for 1200 bps data communication in the United States.

V.22*bis*
CCITT-defined international standard for 2400 bps data transfer over voice telephone lines.

V.32
CCITT-defined international standard for 9600 bps data transfer over voice telephone lines. Also allows for a 4800 bps transfer rate.

V.32*bis*
CCITT-defined international standard for 14,400 bps data transfer over voice telephone lines. Also allows for 12,000, 9600, 7200 and 4800 bps transfer rates.

V.42
CCITT-defined error control protocol. Uses the Link Access Procedure for Modems (LAP-M) to negotiate connections with another modem.

V.42*bis*
CCITT-defined data compression protocol. The theoretical limit for data compression is 4:1. Requires a LAP-M (V.42) connection for operation.

X-ON/OFF
These are start/stop signals issued by two communicating computers to make sure that each is ready to send or receive at the proper time. This protocol is usually built into the communications software, and in most cases you will not be aware that the signals are being sent and received. However, you can generate each signal yourself to stop or start an on-screen scroll when accessing most databases. X-ON is generated by entering a Control/Q, and X-OFF by entering a Control/S.

Xmodem
File transfer protocol that includes automatic error checking and error correction during transfer. Other options for error checking include CRC or checksum and transfer block size.

Ymodem
File transfer protocol that is an extension of Xmodem. It also includes the ability to send multiple files and the name and size of files being transferred.

Zmodem
File transfer protocol that does not wait for positive acknowledgment before sending each block, but does re-transmit unsuccessful packets. ZMODEM is considered one of the most advanced transfer protocols.

How To...

In this section we take a closer look at how to perform some of the most common PowerLine tasks:

Establish a Connection

Before you connect, you first need to decide what you will be using to connect. You can use any of the following methods:

- Modem and phone lines.
- Null modem cable.
- Network.

Once you have determined what you will be using to connect, you then have to decide to whom you will be connecting. Who you connect to will determine many of the PowerLine options you set by using the Hardware Settings and Terminal Settings dialog boxes. For example, to connect to Enable's BBS you can use 2400 baud, no parity, 8 bit word length and 1 stop bit. Check with the system to which you will be connecting.

There are two ways you can connect; by using a listing or by using Manual Dial. Most frequently you will connect by using PowerLine's phonebook where you can create and select a listing. Each listing contains the information needed to connect to a particular information service or computer system. Establish listings for those places you will be connecting to frequently.

Connect by using Manual Dial if you need to call something up quickly, infrequently and without special options set.

Listings

Easy to use and customizable are two buzz words that begin at the PowerLine PhoneBook where you can set up listings specifically designed to be used to connect with whatever information service you'll be using. You can organize where you call by using phone books and listings. Each phone book contains certain types of listings. For example, you might have one phone book that contains all of your listings for connecting to information services. Another might contain all the numbers you use at work.

Each listing contains all the information you will need to connect to another computer system. Also, you can use the listing as an autodialer by selecting the voice option.
Since the best way to learn is by example, we'll first examine the phonebooks and listings already provided by PowerLine.

To display the phonebook:

- From the Top Line Menu select Connect, PhoneBook. The PhoneBook Manager dialog box displays.

The PhoneBook Manager displays the phonebooks already installed by PowerLine:

- *General* - Place any general listings you have here.
- *Services* - Place any listings specific to information services here. We have included listings for Delphi, CompuServe, GEnie, Dow Jones and MCI Mail.
  Take a closer look at Services Phonebook
- *Voice* - Place any listings here you will want to use for voice calls. We have included the customer service numbers for each of our sample services in case you need to contact them.

As you examine this dialog box, notice you can use its features to manipulate how your listings and phonebooks are arranged in addition to creating a dialing queue to dial particular listings in order, edit listings and create new phonebooks and listings.

Looking at a Phonebook

Next, we'll take a closer look at a phonebook. Select Services Phonebook. Each of the sample listings for services displays:

- **CompuServe**. Connect to the CompuServe information service to access news, commentary, current financial reports and analysis, on-line games and shopping, electronic clubs and discussion forums.
- **GEnie.** Connect to General Electric Information Services to access news and commentary, other GEnie users, electronic mail, electronic clubs and discussion forums, games, and shopping.
- **Dow Jones.** Connect to the Dow Jones News Retrieval and Financial Information Service to access the latest financial news.
  Take a closer look at the Dow Jones listing
- **Delphi.** Connect to the Delphi information service to access electronic mail and file base.
- **MCI Mail.** Connect to MCI Mail, a major electronic mail service. From MCI mail you can deliver mail by using electronic telex, printed copy via first class U.S. mail or hand delivery by courier.

Each of these listings contains all of the information PowerLine will need to establish a connection. If you selected Information Service Setup during installation, PowerLine has already included the local node for the information service. Otherwise, you'll have to include it. If you use one of these services for the first time, you'll be prompted to include your user id and password for the service. See Use Sample Listings for a detailed description.

A Closer Look

Let's take a closer look at the Dow Jones Service listing. Select the Services phonebook and select Dow Jones. The Phone Entry dialog box displays. This dialog box contains all of the information PowerLine will use to connect to Dow Jones.

This box contains the name of the service, your user id, your passwords and the service's phone number. Note that all phone numbers are included as if they were long distance. Before connecting, PowerLine will check to see your local area code as set in the Dialing Settings dialog box and make the proper adjustments, so if you suddenly change area codes you only have to include your new one once without altering your listing numbers. In addition to the listing basics you can truly customize each listing by associating the following types of information with a listing: (Many of these settings are indicated by the information service, in our case Dow Jones, so you need to check the documentation for the service or BBS to which you want to connect.)

- **Hardware.** Hardwarespecifications required by Dow Jones.
- **Terminal.** Terminal Settings required by Dow Jones.
- **Toolbar.** Custom or default toolbar .
- **Menu**. Custom or default Top Line Menu. We have designed a special Top Line Menu to include tasks you would perform frequently when using Dow Jones.
- **Script.** Login script that contains the keystrokes you need to press when logging onto Dow Jones.
- **Outgoing Filter Map.** The Outgoing Filter Map "exchanges" a typed character to another character automatically..
- **Incoming Map.** The Incoming Filer Map "exchanges" incoming characters with pre-defined substitutes.
- **KeyBoard Map.** Custom keyboard map. We have designed a special keyboard map that maps the function keys to tasks you perform frequently when using Dow Jones.
- **Scratch Pad.** Custom Scratch Pad for storing "scribbles" of information.

If you choose not to associate any custom settings, PowerLine will use its default. You can also edit any of the listing settings from here as well as from the Top Line Menu.

Just to give an idea of what all of this looks like when you use it, click on Open. The PowerLine Dow Jones window displays.

When you have finished viewing the window return to the default PowerLine window by selecting Connect, Phonebook, General, Default Phonebook Listing and Open.

Using Sample Listings

If during installation you selected information service setup and included the appropriate local node, all you need to do to use any sample listing is click on the listing and go. PowerLine will prompt you for your user id and password at the appropriate time.

To use a sample listing:

1. Start PowerLine.
2. From the Top Line Menu select Connect, PhoneBook.
3. From the PhoneBook Manager dialog box select the Services Phonebook and the appropriate sample listing.
4. Select Dial. Powerline displays the terminal window, echoes modem commands back to screen and displays the Dialing Status dialog box. This dialog box displays dialing progress.
5. PowerLine connects to the service. If it is your first time connecting to the service, PowerLine prompts you for your user id and password. Consult the information service instructions for information on using it and disconnecting.

Manual Dial

Sometimes you just need to call somewhere once, quickly and without concerning yourself with any options. Use Manual Dial. From the Top Line Menu select Connect, Manual Dial and enter the appropriate information.

The most common telecommunication method. Make sure you have selected the proper Modem Settings and Dialing Settings. Set the appropriate comm-port to Modem by using the Port/Modem Setup dialog box.

A special cable you can use to transfer information directly between the serial port on two PC's. Set the appropriate comm-port to Direct by using the Port/Modem Setup dialog box. Set the baud rate to the top speed supported by both systems by using the Hardware Settings dialog box. Transfer files by using the Send File/Receive file dialog boxes or PowerLink.

Connect through a network modem pool. See your network administrator for information on the special drivers you will need. Set the appropriate comm-port to Int14 by using the Port/Modem Setup dialog box.

Transfer Files

When you are communicating, much of that time will be spent sending and receiving files. This task is as easy as copying to disk when you use PowerLine's Send or Receive dialog boxes. When you are ready to send or receive, from the Top Line Menu select Connect, Send or Receive a file. Specify the file name(s), protocol and select OK.

Transmit and Receive Files

To transfer files between computer systems perform the following general steps:

- Connect with the other computer system.
- Tell the other computer you wish to upload or download file(s) and select a transfer protocol. (Be sure you select a protocol supported by PowerLine).
  Protocol is a standard way of regulating data transmission between computers. A protocol allows you to transmit any type of file transparently, without having to watch the data in each file scroll on your screen. It is generally determined by the system to which you are connecting and both systems must use the same protocol. PowerLine supports the most popular protocols including Xmodem; ASCII send/receive; Xmodem CRC; Xmodem 1K; Ymodem batch; Zmodem; Compuserve B+; and Kermit. Use the Protocol Setup dialog box to customize the way in which protocols operate.

Next, tell the other computer the file name(s) of the file you wish to transfer and wait for permission to begin the transfer.

- From the Top Line Menu, select Connect and Send a File or Receive a file. Select the protocol you and the other system will use to transfer the information. Protocol is a standard way of regulating data transmission between computers. A protocol allows you to transmit any type of file transparently, without having to watch the data in each file scroll by on your screen.

  Sending files using ASCII and Xmodem protocols
  Sending files using Ymodem Batch, Kermit, and Zmodem protocols
  Sending files using the Compuserve B+ protocol

Protocols

You can select the following protocols:

- **ASCII**. Sets no protocol and allows you to transmit between systems that do not support any of the other protocols. ASCII protocol is normally used with ASCII text files, but you can also use it for other, non-binary, data.
  If you use ASCII, you and the other user should activate the Xon/Xoff flow control by using the Hardware Settings dialog box to avoid loss of information during transmission. If you do not use Xon/Xoff, the receiving system may receive data before it is ready, and the data will be lost. The risk of loss rises as transmission speed (baud rate) rises. Do not use ASCII for files with binary data.
- **Xmodem**. An error-checking procedure that ensures accurate transmission and is widely supported. Any of the Xmodem protocols are the most widely supported by personal computers.
- **Xmodem CRC**. Another type of Xmodem error-checking procedure that ensures accurate transmission and is widely supported. Since CRC-Xmodem is more accurate than Xmodem-Checksum, we recommend that you use this protocol instead of Xmodem-Checksum if it is supported.
- **Xmodem 1K**. Another type of Xmodem error-checking procedure that transfers data more efficiently since it handles larger blocks of data. The 1K-Xmodem protocol transfers blocks of data in 1K sections instead of 128 byte sections used by other Xmodem protocols.
- **Ymodem batch.** An extended version of Xmodem that can quickly handle batch transfers by using larger blocks of information. You can use this protocol when you want to transfer multiple files. Since this protocol does use large blocks of information, it may not be the best protocol to use if your transmission line has a lot of noise. The noise interference may cause loss of large sections of data. If the other system does not support Ymodem batch and you use it, PowerLine will automatically try CRC-Xmodem, 1k-Xmodem, and Xmodem-Checksum to find a protocol that works.
- **Ymodem G.** Sends data in a continuous stream without the typical wait for acknowledgment at the end of each data packet.
- **Zmodem**. One of the most advanced and efficient protocols. Sends each block without a positive acknowledgement after each block is sent. Re-transmits unsuccessful packets. If the transfer is interrupted at any time it can be resumed without resending information. Allows you to send multiple files.
- **Kermit**. An error-correcting file transfer protocol used by many mainframes and microcomputers. Enable's Kermit protocol is a subset of the Kermit data transmission program created at Columbia University. It is designed to let two Kermit programs communicate directly with each other in order to transfer files. Kermit lets you transfer data as 7-bit and 8-bit words. One benefit is that it can transfer data as 8-bit words even if both computers use 7 bits as a word size. You can also use Kermit to transfer multiple files.
- **CompuServe B+**. A file transfer protocol that is used by the CompuServe Information service.

ASCII and Xmodem Protocols

To transmit a file:

1. From the Top Line Menu, select Connect, Send a File. The Send File dialog box displays.
2. Specify the file you want to send, the protocol you want to use and if you want to display the PowerLine window as an icon during transfer. When you select OK, PowerLine begins transmitting the file and displays the status in the Send Using dialog box. If any errors occur during transmission, PowerLine will alert you and allow you to perform the appropriate action.

To receive a file:

1. From the Top Line Menu, select Connect, Receive a File. The Receive File dialog box displays.
2. Specify the name under which you want to receive the file, the protocol you want to use and if you want to display the PowerLine window as an icon during transfer. When you select OK, PowerLine begins receiving the file and displays the status in the Receive Using dialog box. If any errors occur during the transfer, PowerLine will alert you and allow you to perform the appropriate action.

Ymodem Batch, Kermit and Zmodem Protocols

To transmit a file:

1. From the Top Line Menu, select Connect, Send a File. The Send File dialog box displays.
2. Specify the file you want to send. These protocols allow you to specify multiple file transfers. Enter the appropriate wild cards and select the protocol you want to use and if you want to display the PowerLine window as an icon during transfer. Select Wildcard Send, PowerLine begins transmitting the file and displays the status in the Send Using dialog box. If any errors occur during transmission, PowerLine will alert you and allow you to perform the appropriate action.

To receive a file:

1. From the Top Line Menu, select Connect, Receive a File. The Receive File dialog box displays.
2. Specify the name(s) under which you want to receive the file and the protocol you want to use. These protocols allow you to specify multiple file transfers. Enter the appropriate wild cards and select the protocol you want to use and if you want to display the PowerLine window as an icon during transfer. When you select OK, PowerLine begins receiving the file and displays the status in the Receive Using dialog box. If any errors occur during the transfer, PowerLine will alert you and allow you to perform the appropriate action.

CompuServe B+ Protocol

If you are using CompuServe B+, select Connect, Send a File/Receive a File and set the appropriate options. When you select OK the the CompuServe system automatically takes over the transmit/receive procedure. CompuServe uses a variety of commands to perform file transfers. See CompuServe for information on what commands it requires.

Emulate a Terminal

You can make your PC act like a terminal for a mainframe computer in order to communicate with the mainframe. By imitating a terminal, you can take advantage of features such as text editors offered by the mainframe or run a large computer program.

You can emulate a TTY, DEC VT100/102, VT52, DEC VT220/320, or ANSI (X364-79) terminal. PowerLine automatically converts or filters the control sequences used by these popular terminals.

To use terminal emulation:

1. Create a terminal settings file. Check with the system you will be connecting to for setting information.
2. Associate the Terminal settings with the listing you will be using to connect with the other computer. After you are connected, your PC will process the commands exactly as the terminal would. See Appendix J: VT Emulation and Numeric Keypads for further description on using terminal emulation.

Use PowerLink

If you need to get information from your home or office PC quickly and easily, you'll find PowerLink to be the solution. Use PowerLink to access another PC running PowerLine and find files on that PC remotely, and send or receive those files. No more carting armloads of disks between work and the office and trying to guess which ones are most current.

To access PowerLink:

- From the Top Line Menu select Connect, PowerLink.

Two of the most common PowerLink scenarios are bi-directional sending and receiving with chat, and file exchange with an unattended PC.

Bi-directional Send and Receive

Two PowerLine users may use PowerLink to connect and bi-directionally exchange files and chat messages. To begin the exchange, one user sets PowerLine to Wait for Call (via the Connect menu); the other dials and connects.

In the PowerLink dialog box, click on Send Path and select drive, directory and search criteria to display the files you want to send in Local Files. As you scroll through Local Files, the size of the highlighted file displays in File Size.

To send a file, double click on a file name in Local Files. The file name displays in Send List; Transfer Status indicates the progress of the file transfer. The left icon (Send Status) changes to "BUSY" when sending files and the middle icon (Connect Status) displays the transmission rate.

The names of incoming files display in Remote Files on the remote (receiving) PC. File size and transfer progress are also displayed. The right icon (Receive Status) displays "BUSY" when receiving files.

To initiate chat with another PowerLink user with whom you are connected, press the Send Message button and enter text in the Outgoing message area. Press Enter to send the message. Your message will display in the Incoming message area on the remote PC.

Exchange with an Unattended PC

You can configure PowerLink to allow access to files on an unattended PC. Transfer files quickly and easily between office and home.

To set up the unattended PC, first set the Wait for Call option in the PowerLine Connect menu. Select the Setup button from the PowerLink dialog box and specify the directory and files you will make available to callers. (Use the Browse button to select or create new paths.) Specify a password if desired and select a PowerLink Exit option.

Callers to the unattended PC may dial, connect, and activate PowerLink from the Connect menu. The PowerLink dialog box displays with the message "Connected to PowerLink Version 1.1" and prompts for a password, if necessary.

To receive files from the unattended PC, click on Remote Path and specify a path; Remote Files displays available files. As you scroll the list of available files, the size of the highlighted file displays in File Size. Double click on a file name to place it in the Receive List and initiate the download to your computer. Transfer Status displays the progress of the file transfer.

To send files to the unattended computer, click on Send Path and specify a path on your PC. Double click on a file name in Local Files to place it in the Send List and send it to the unattended computer.

Littera scripta manet.

Exegi monumentum ære perennius.

(Or so I hope.)

-Psyclops

Internal Help Version: 4.5.93d