PEP: An Overview

The problem

You have a computer. You've had it for some time, and a lot of very valuable data is stored on its diskettes.

You've also just acquired an IBM PC.

You want to transfer your data to the PC. As soon as possible, and as painlessly as possible.

How PEP solves it

PEP is a simple, inexpensive, and universal technique for reliably and easily transferring data from virtually any computer to an IBM PC or compatible. While almost any application is running on the transmitting computer.

PEP stands for "Printer Emulation Package." Briefly, PEP will make your PC look like a printer to whatever computer is attached to it.

It's a simple idea, easy to understand, and it works beautifully. With PEP, if an application can print, then it can transfer data to your PC. All by itself, without the need to run (or learn about) another piece of software.

Some details

To be more specific, PEP is intended to make your IBM PC look like a printer which is attached to your other computer using an RS232C interface

Unlike most printers, however, your PC is equipped with disk drives. PEP can store the data which your PC receives on diskette or on a hard disk, so that you may conveniently access it on your PC. In fact, if you want it to, PEP can simultaneously receive the data, store it on diskette, display it on your monitor and print it on a printer (the ordinary kind) attached to your PC.

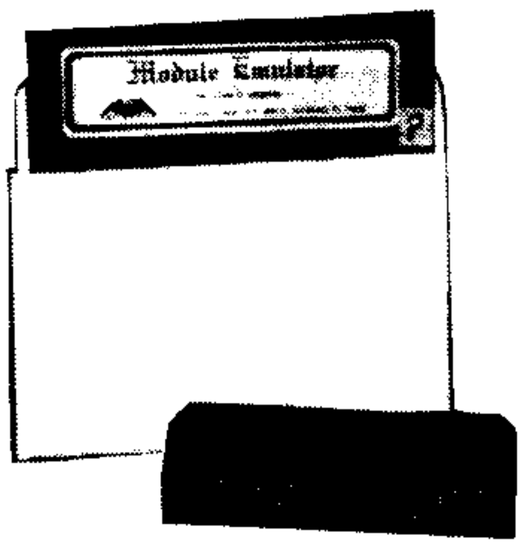
DATA CONVERSION

P E P

PRINTER EMULATION PACKAGE

A SIMPLIFIED UNIVERSAL DATA COMMUNICATIONS / DATA CONVERSION PROGRAM FOR THE IBM-PC AND COMPATIBLES

TURN A FAT BOX FULL OF MODULES INTO A FEW SKINNY FLOPPIES!



If you are like most TI owners, you have accumulated quite a few pieces of cartridge software over the last few years. The only problem you have found, most likely, is where to put them all! Even with specially designed storage systems, TI modules take up a lot of real estate on your computer table — and who ever had room to spare there? Well, a solution is at hand. . .the exciting new 6000+ module is the last module you'll ever need. In conjunction with the Module Emulator software, it lets you dump your cartridges into disk format and store many cartridges on a single floppy disk.

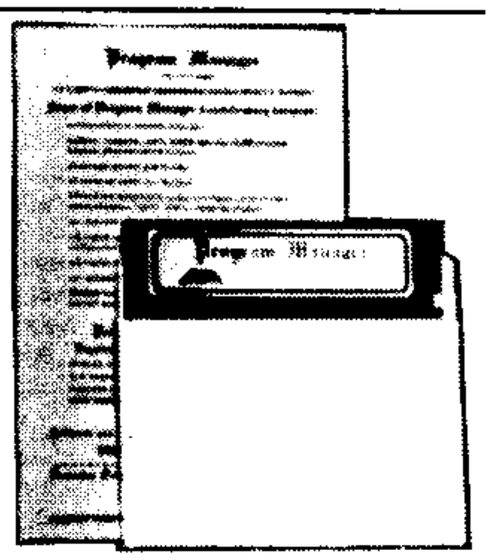
The 6000+ module is a real breakthrough — while hackers have been converting cartridges to disk format for years, the disk programs required manual "patches" or changes to the code to make them work. This is because module software looks for memory locations that are not located in the computer RAM or the memory expansion RAM — these locations are only in the cartridge. The 6000+ module provides these locations, so the software works without modification.

Imagine the convenience of having all your favorite education programs on one disk! Or of being able to put all of your game cartridges in the attic because they're on one handy floppy! Here's what you'll need to accomplish this: a 6000+ module, the Module Emulator Software, a disk system, 32K or larger memory expansion, and a cartridge expander (see page 40). Disk versions of module programs must be run with the 6000+ cartridge. The Module Emulator will work with virtually all modules: an extensive testing program has proved its performance with over 100 modules! It will not work with Extended BASIC, Plato, MBX cartridges, Personal Record Keeping, Statistics, Disk Manager, and Tax Investment Record Keeping.

The 6000+ is a truly powerful innovation — in addition to the Program Manager software described below, we expect to see additional valuable utilities written for it. So don't delay — start saving time and space and preventing needless wear and tear on your computer today!

42457 6000+ Module plus Module Emulator Software — Sug. Retail $69.95 — **$59.95**

SUPER ORGANIZER FOR YOUR DUMPED MODULES!



So now you've got your modules all neatly stored on a few disks — how can life get still easier? With the Program Manager, you can create a super menu system for accessing hundreds or even thousands (if you can find that many) of assembly language programs. When you select the program you want, the Program Manager will automatically find it, load it, and run it for you. You design your own menus and menu levels — you can have, for example, a broad menu category — say, education — and break that down into sub-menus for spelling, math, etc. It will automatically search up to 5 disk drives in any order you define and in any configuration (single side, double side, single density, double density). It will even work with Myarc's RAM disk function.

Program Manager will organize, load, and run all of your assembly language programs, including modules that have been dumped to disk and other assembly language programs that you would normally have to load with the TI Editor/Assembler module. Program Manager requires the 6000+ module, a disk system, and a memory expansion of 32K or greater.

42461 Program Manager, Disk — Sug. Retail $25.95 — **$21.95**

PEP

ORDERED 10 SEPAL
EXPECT TWO WEEKS
TO DELIVER

—Transfers files to IBM PC

—Works with any computer, including TI 99/4A

From A Happy Customer. . .

Re: "PEP" (Printer Emulation Package)
I have just transferred all my word processing files from my Texas Instruments TI 99/4A (which were done with "TI Writer") to my new AT&T 6300 Personal Computer (on which I am using "Word Perfect") using "PEP". Everything went without a single hitch, which, based upon my experience with computers, is a miracle. However, I really think it is due to the product and the manual, which is clear (amazingly so) and definitive.

This product has saved months of retyping of forms that I use daily in my law practice, for which saving of time and work both my secretary and I thank you profusely.

—Illinois

PEP Printer Emulator Program is a utility that runs on your IBM PC and turns it into a "printer" for other computers. The PC accepts the input through its RS-232 port, and can display it on the screen, print it on a printer, and/or save it to disk. PEP will accept data at a very high rate of speed — 960 characters per second or about 50K per minute. Even lengthy lists or documents can be "printed" to the IBM PC in a snap. The new file created by the IBM PC is an ASCII file for easy manipulation by commercial or user written software.

The uses for PEP are many. Users who have created lengthy mailing lists, word processing documents, or spreadsheets on the TI or other computers may want to transfer them to an IBM PC for use in sophisticated programs like 1-2-3, Wordstar, etc. A user with limited time available on a PC might use another computer to actually input data or documents and then transfer them to the IBM PC. (It's almost like getting a second IBM PC for free!) The time saved by using PEP instead of retyping documents or data could pay for the program in just a few days, not to mention the prevention of keying errors.

Here's what you need: a computer (TI or other) with an RS-232 output, an IBM PC with at least 128K of memory (256 preferred), and an RS-232 output, DOS 2.0 or 2.1 (or higher) for the IBM, and a connecting cable. The documentation describes the cabling requirements so you may make your own or provide specifications to a cable maker. The cable listed below fits most configurations for connecting the TI 99/4A RS-232 output to the IBM PC; check with our Customer Service Department to confirm that it is the right one for your setup.

Package includes TI and IBM PC software plus complete and easy to use documentation. Runs on IBM PC and many compatibles.

38131 Disk, IBM PC — **$59.95**
38145 TI/IBM Cable — **$19.95**

TENEX Computer Express™

Our Customers Speak:

"I would very much like to thank you for your fast service in delivering my two software packages. I had received them just in time for school and I was very happy to confirm that they were exactly what I wanted."

—New York

WE WANT SATISFIED CUSTOMERS!

All items in our catalog are new, first quality, and have full manufacturers warranties. Should you need specific warranty information, please contact our Customer Service Department.

We guarantee our products to be free from defects and to be in complete working order. If any item is found to be defective, it may be returned within 30 days of receipt for prompt repair or replacement. Please contact our Customer Service Department for a Return Authorization.

ORDER TOLL-FREE 1-800-348-2778 INFORMATION 1-219-259-7051 ORDER STATUS HOTLINE 1-219-259-7053.

FALL 86 CATALOG

Intelpro
13 Saratoga Drive
Kirkland, Quebec, Canada
H9H 3J9

Dear customer,

Thank you for purchasing PEP!

You'll need a cable to connect your PC to your remote computer. This is explained in the appendices of the manual, which you should read very carefully.

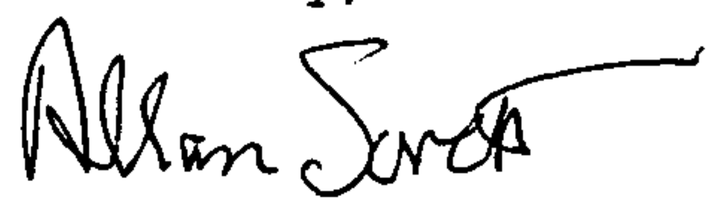
PLEASE READ THE FILE "READ.ME" ON THE PEP DISKETTE. IT EXPLAINS THE CONSIDERABLE NUMBER OF ADDITIONS TO THE PEP PACKAGE, AND CORRECTS SOME UNFORTUNATE TYPOGRAPHICAL ERRORS IN THE MANUAL.

Please also note our new address, which takes effect on June 30, 1986. Enclosed, you will find four address stickers on which this new address is printed; please use one of these to correct the mailing address on the reverse side of the title page of the manual. As well, if you are sending us your owner registration after June 30, 1986 please use an address sticker on the Owner Registration Form at the front of the manual (it's erroneously labeled "Product Information"). As well, please place one sticker on each of the Product Information Forms at the back of the manual.

Until June 30, our mailing address is as printed in the manual, and our telephone is 1-514-656-8798. We'll be without phone service between June 30 and about July 10; after that date, you can reach us by calling Montreal operator assistance at 1-514-555-1212, and asking for my personal number.

Lastly, please do mail the Owner Registration form to us! This form provides us with the information we'll need if we're to communicate with you regarding improvements or additions to the PEP resources, and additional software products.

Yours truly,



Allan K. Swett
President, Intelpro

Please affix sufficient postage

TO : INTELPRO
13 Saratoga Drive
Kirkland, Quebec, CANADA
H9H 3J9

Fold here

Product Information : PEP

Tape Please do NOT staple Tape

PEP

PRINTER EMULATION PACKAGE

A SIMPLIFIED UNIVERSAL DATA COMMUNICATIONS / DATA CONVERSION PROGRAM FOR THE IBM-PC AND COMPATIBLES

Table of Contents

Preface

Welcome to PEP ("Printer Emulation Package"), the world's simplest and most understandable communications/ data conversion software.

PEP consists of this owner's manual and the programs and data on the PEP diskette.

Diskette organization

The PEP diskette consists of the program files

PEP.COM
PEP.OV

and a readable file entitled

READ.ME

READ.ME is to be considered a continuation of this manual. It may be printed on your screen using a DOS command such as

TYPE READ.ME

when the PEP diskette, or a copy of it, is placed in the default drive. It will give further instructions, or updated information, or corrections to this manual. As well, it will describe any additional files which have been placed on the PEP diskette.

READ.ME is an extremely important extension to this software package, and you should be certain to read its contents!

Manual organization

This manual is organized as follows:

"PEP: An Overview" is a nontechnical introduction to PEP. It includes a summary of the problems which PEP has been designed to solve, the method in which PEP solves them, and the basic considerations involved in using PEP.

Chapter 1, "Getting Started," lists the minimal hardware and operating system which are required to use PEP. This chapter also indicates how to make a backup copy of PEP and a "working copy," and how to start PEP.

Chapter 2, "Telling PEP What It Needs, and What You Want," explains how to supply PEP with two types of information. The first type involves the way that your "remote computer" will be communicating with your PC. The second type of information involves the disposition of the data which enters your PC. As well, it provides an insight into how PEP works.

Chapter 3, "Receiving Some Data Files," provides a first experience in using PEP to receive data from your remote computer.

Chapter 4, "Putting PEP to Work," explains the full range of capabilities of PEP, and how best to exploit them.

Chapter 5, "Error Conditions," summarizes the problems which may arise as you use PEP. It lists the error messages which you may encounter, provides an interpretation of each, and indicates how to proceed after an error has occurred.

Appendix A, "Interfacing a Remote Computer to Your PC: What PEP Requires," explains how PEP uses the communications facilities of your PC.

Appendix B, "Interfacing a Remote Computer to Your PC: Making a Cable," explains how to obtain a cable to connect your remote computer to your PC.

Appendix C, "Interfacing Your PC to a Modem or to a TI-99/4A," explains how to connect your PC to a standard modem or to a Texas Instruments TI-99/4A Home Computer.

Appendix D, "Receiving COMPANION Printouts," explains how to use PEP to receive data files from a particular word processor running on the TI-99/4A.

Appendix E, "Technical Notes," lists some technical considerations regarding running PEP on computers other than an IBM PC, and regarding PEP's use of the operating system.

Please don't be too intimidated by the size of this manual. It's extremely complete and quite wordy, but very easy to read and to use.

If you're intent on learning quickly about PEP, I suggest that you at least read Chapters 1 and 2 before running the program. As well, you should read as much of the Appendices as you need, in order to obtain the correct cable.

Allan Swett
Montreal, September 1985

PEP: An Overview

The problem

You have a computer. You've had it for some time, and a lot of very valuable data is stored on its diskettes.

You've also just acquired an IBM PC.

You want to transfer your data to the PC. As soon as possible, and as painlessly as possible.

How PEP solves it

PEP is a simple, inexpensive, and universal technique for reliably and easily transferring data from virtually any computer to an IBM PC or compatible. While almost any application is running on the transmitting computer.

PEP stands for "Printer Emulation Package." Briefly, PEP will make your PC look like a printer to whatever computer is attached to it.

It's a simple idea, easy to understand, and it works beautifully. With PEP, if an application can print, then it can transfer data to your PC. All by itself, without the need to run (or learn about) another piece of software.

Some details

To be more specific, PEP is intended to make your IBM PC look like a printer which is attached to your other computer using an RS232C interface.

Unlike most printers, however, your PC is equipped with disk drives. PEP can store the data which your PC receives on diskette or on a hard disk, so that you may conveniently access it on your PC. In fact, if you want it to, PEP can simultaneously receive the data, store it on diskette, display it on your monitor, and print it on a printer (the ordinary kind) attached to your PC.

And PEP uses a "buffered interrupt-driven communications driver," so that it can receive data from the remote computer just as fast as the remote computer can send it. (In other words, PEP makes your PC behave like a printer with a "print spooler.")

PEP will store separate data transmissions as separate diskette files. PEP does this by recognizing a five-second pause between transmissions as an "end of transmission" indicator. This offers you the convenience of transmitting a sequence of distinct data files to your PC and having them stored as distinct diskette files — automatically, without touching the PC, and without leaving the keyboard of the other computer.

Useful? Well, one of our first customers transferred 550 distinct data files to his PC-XT. While PEP was automatically saving these as 550 distinct files on the PC, he didn't have to walk from one computer to the other even once.

If you have a large number of files to transfer, you'd probably prefer to have all of them transferred automatically, without having to attend to either of your computers until the entire transfer is done. PEP will make this easy. (Our early customer did his file transfers this way, one diskette at a time.)

As well, PEP will allow you to disconnect, reconnect, crash, turn on or turn off (or even change) the sending computer between data transmissions. PEP does this without difficulty (or manual intervention at the PC). Even if the sending computer tends to transmit "junk" while you're making adjustments, PEP is smart enough to ignore the glitches.

How to use PEP

PEP operates in two phases.

To start PEP, you'll enter the word PEP at the keyboard of your PC, in response to the system prompt (usually A>). PEP will load, and commence Phase I of its operation.

In this phase, PEP will present you with a brief questionnaire. The questionnaire will request some information about the kind of communications linkage you have between the two computers and the way you want to handle the incoming data. You'll tell PEP what you want.

Then PEP will enter Phase II, turning your PC into a "printer."

You may then leave the PC and start to work on the other "remote" computer. You may load and run virtually any application on the remote computer, and then request the remote to print through one of its RS232 ports. Whatever goes out, be it from a word processor or a spreadsheet or a compiler, will enter the PC, and will be treated as you specified in Phase I.

At the remote computer, you can change applications (say, from a spreadsheet to a word processor to an operating systems utility) and print from each one, all without touching your PC. And as mentioned before, between data transmissions you can turn the remote on or off, adjust the cabling, or even use a different computer as the remote, again without any adjustment whatsoever at the PC.

Limitations

Any limitations?

Only this: for your convenience, PEP tries to do a good job of recognizing separate transmissions (we'll call these "data packets") and storing these in separate diskette files on the PC. It does this by recognizing a pause between transmissions, whose duration is usually five seconds. This "end of packet indicator" may be adjusted during Phase I if you wish.

So, the application you run on the remote computer should print rather continuously (like most word processors), AND you should wait a few seconds before you transmit the next data packet.

To help you do this, the PC will beep when it has recognized an end of packet pause, at the end of the pause. This will tell you that you may immediately transmit the next data packet. The PC's display screen will always let you know what's happening, should you not hear the beep.

You can easily reenter Phase I from Phase II (one keystroke at the PC keyboard will do it), so that you can reinstruct your PC as to how to handle the incoming data, or indicate a different end of packet pause.

And . . . don't worry about the pauses. PEP will store everything you transmit in some diskette file, period. If you make a mistake, your diskette files will just be slightly irregular.

Why use PEP?

There are a lot of computers out there, and an awful lot of applications software. PEP works the same with all of them, including those which haven't been invented yet, so the brief time you spend learning how to use PEP will be well invested.

Even if your other computer is extremely sophisticated and both it and the PC are networked to a mainframe with impressive communications capabilities, and you're a brilliant systems programmer and you already understand the whole installation, PEP may still prove to be the transmission vehicle of your choice. Because in addition to being simple and reliable, it's really, REALLY FAST !

Almost any applications software, no matter what its origins or purpose, can at least print something. It is quite likely that it can print through RS232 cables, since a great many printers are hooked up this way. In many cases, an application can print exactly what you want to have on your PC. Splendid — just let the application itself print directly to your PC, using PEP.

If this isn't quite the case, your application may be saving what you want on diskette. No problem, just "copy" the diskette file to your RS232. (If the remote computer has a minimal operating system, a tiny bit of programming might be required here, maybe twenty lines of BASIC.) And, in the rare instance that your application doesn't print or save exactly what you want, you can transfer the best it can do to your PC using PEP, and then manipulate it on the PC.

PEP is well engineered. It's flexible, it's durable — meaning that we don't think it will crash — and it's extraordinarily quick. The crucial parts are finely crafted in assembly language, and of course the RS232 communications routines are interrupt-driven as well.

PEP is very, very simple to operate and understand. Our goal in designing PEP, including using the printer metaphor, was to make it just as easy to understand as possible. We think you'll find it much easier to understand than most printers.

Chapter 1.

Getting Started

1.1 Minimal requirements

PEP will undoubtedly run on a number of MS-DOS machines. In this manual, however, we'll assume that you're running PEP on an IBM PC.

PEP consists of this instructional manual and a diskette. To run PEP, you'll need the PEP diskette and

- an IBM personal computer ("PC")
- a diskette drive (single or double sided)
- 192K bytes of memory, 256K recommended
- a monitor
- a Disk Operating System diskette, Release 2.0 or higher ("PC-DOS" or "MS-DOS")

As well, you'll certainly want

- some DOS formatted diskettes for storing data
- a "remote" computer, to send data to your PC
- a cable to connect the two computers

1.2 The remote computer and the cabling

PEP is designed to operate with as many "remote" computers as possible, ranging from a battery-operated, hand-held portable to a large mainframe. Your remote computer should be able to send characters though an RS232C communications line. These characters should be encoded using the ASCII standard — if not, PEP will still hand the incoming data to your PC; you'll have to translate the data later. All popular microcomputers use the ASCII standard.

Forgive some technicalities:

(A) Your "remote" compuper should be able to transmit RS232 data at one of the rates 300, 600, 1200, 2400, 4800, or 9600 baud, using either seven or eight data bits, with one or two stop bits. The ability to transmit in any one of these ways will suffice.

(B) The remote should sample its DSR ("Data Set Ready") or CD ("Carrier Detect") pin, or both, before transmitting data.

(C) It would be preferable if the remote computer asserts (or can be made to appear to assert) control over its RS232 pin DTR ("Data Terminal Ready"). It should keep this pin continuously "on" while the remote computer is turned on, or is in the mode in which you'll be using it to transmit data. PEP will still function well if this is not the case, but will not detect remote disconnections.

The IBM PC does all of the above; your remote computer probably does also. Don't worry too much about (B) and (C). In fact, (B) is a certainty, since otherwise you'd never be able to connect your remote computer to a printer. (Why? The printer MUST be able to prevent your remote from sending data, while it's in the process of printing and its print buffer is full.)

To connect the two computers, you'll need an appropriate cable. See Appendices A, B, and C for information regarding the cabling, especially as pertains to the RS232 signals which PEP will interpret and control.

Using the appendices, and a bit of patience, we believe that you'll successfully produce a cable conforming to PEP's requirements, regardless of the computer you're attaching to your PC. For most remote computers, this will be a "standard null modem cable." Modems, the TI-99/4A Home Computer, and other "DCE" devices will use a straight through cable.

1.3 Make a backup and a working diskette

Prior to using PEP . . .

Make a backup copy of the PEP diskette, using the DOS DISKCOPY utility. We assume you know how to do this, and in fact we'll assume that you're fairly familiar with the PC-DOS (or MS-DOS) operating system. PEP is not in any way copy protected.

If you have a hard disk, you should next copy all of the files (from your copy of the PEP diskette) to your hard disk. Put them all in the root directory of the hard disk. ← NOTE

If you'll be running PEP from a floppy diskette rather than a hard disk, first format one, using the /S option so that the system files will be transferred to your floppy. You can use the single sided option /1 if you wish, since PEP's files are rather brief. Copy all of the files (from your copy of the PEP original) to your "working" diskette, the one you've just formatted. Put them all in the root directory.

To run PEP, you'll need at least the files PEP.COM and PEP.OV on your working diskette. Inclusion of other files from the PEP original is optional, so you may erase these from your working diskette if you wish. But . . . don't forget to read the file READ.ME, it's a very important part of this software package.

You may then put away your original PEP diskette, and your copy of it, in a safe place.

1.4 Starting PEP

We'll assume that you're running PEP from a floppy. If you're using a hard disk, you'll probably infer how to use PEP from the description which follows.

In fact, since we're assuming that you know a bit about MS-DOS, you probably already know how to start PEP more easily than we're about to suggest, so what follows is intended primarily for novices.

Place the PEP diskette in drive A. (We mean your "working diskette," the one with the system files and the PEP files in its root directory.) Start DOS by either switching on the PC or (if your PC is already on) holding down the Ctrl, Alt, and Del keys simultaneously.

Your DOS may now require you to enter the date and the time. (To "enter" an expression means to type it and then press the enter key. On a standard IBM PC, the enter key is the one with the angled arrow, three keys to the right of the P key, and "down half a key.") After you've done this, you'll see the DOS prompt

A>

on your screen. Type PEP (the letters will appear just to the right of the prompt) and then press the enter key. PEP will begin loading, and will immediately display

PEP loading...

PEP consists of two programs, one which automatically loads the other. The first one (called PEP.COM) prints the above message, and then looks for the other program (PEP.OV). PEP.COM will look for PEP.OV first in the current directory, and then in the root directory, of the "default" drive. (That's drive A in this case.)

If PEP.COM finds PEP.OV, then it loads this program and begins its execution. You'll then see PEP's main title screen. Since the PEP files are in the root directory of the diskette in drive A, this is just what should happen.

Otherwise, meaning that PEP.COM can't find PEP.OV, you'll get an error message. See Note 4 at the end of this chapter if this happens.

If you're trying to run PEP on a PC with less than 192K (or with so many installed utilities that your PC doesn't have much memory available), you might get an error message. See Chapter 5 if this happens.

1.5 Terminating PEP

You may terminate PEP at any time by pressing Ctrl-T. A note to this effect appears at the bottom of PEP's screen.

1.6 Command line options

PEP allows you to specify a number of options as part of the "command line" which loads it.

If you're following this documentation, you've already used a command line. That was the expression "PEP" that you entered in response to the DOS prompt A>. Any expression which you enter next to A> is called a command line.

Buffer size

Entering the command line

PEP /B = 16

will load a more compact version of PEP. This is appropriate if you're using a PC without much installed memory.

PEP allows a general option of the form

/B = nn

to limit the size of the "communications buffer" which is set aside as PEP is loaded. Ordinarily PEP uses 32K bytes for this buffer, but you may specify a smaller buffer by allowing nn to be one of the values 1, 2, 4, 8, 16, or 32. Here nn will be used as the number of K bytes the buffer will occupy. (One K is exactly 1024.)

If you want PEP to use a small communications buffer, then this "command line option" must be entered as part of the command line you use when you start PEP. Use at least one space between the command line options and NO SPACES WITHIN THE OPTION. For example, if you have a very small PC you might want the buffer to occupy only 1K, that is 1024, bytes of its memory. You'd use the command line

PEP /B = 1

Send — No Send pin selection

PEP allows you to use a command line option to specify which of the RS232 pins the remote computer samples before transmitting data. PEP will work if the remote samples either its DSR ("Data Set Ready") or CD ("Carrier Detect") pin, or both, before transmitting data.

Ordinarily PEP will assume that the remote is sampling the CD pin, which is normally wired to the PC's RTS ("Request to Send") pin via the cable. In other words, PEP will be giving "send — no send" advice through the PC's RTS pin.

If your remote is sampling the DSR pin (and not sampling the CD pin), then you should include the option

/D

in the command line. In this situation, PEP will give "send — no send" advice through the PC's DTR ("Data Terminal Ready") pin, which is customarily wired to the remote's DSR pin.

The option /D is provided only for very unusual situations. In fact, it is provided specifically so that you may attach a TI-99/4A to your PC using a straight through cable. So, if you're in any doubt, DON'T USE THE OPTION /D. If you're using a standard null modem cable to connect your computers, it's almost certain that you shouldn't use /D.

The "/NOTEST" option

If the remote computer does not consistently assert control over one of its RS232 pins to indicate that the RS232 hardware is available, you should run PEP with the command option

/NOTEST

This instructs PEP to ignore all RS232 control signals sent by the remote computer. (Actually, PEP normally only monitors the PC's DSR pin, which is customarily wired to the remote's DTR pin.)

If you use /NOTEST, then PEP will not know when there is a disconnection at the remote. You should therefore refrain from turning the remote on or off, or disconnecting the cabling, while PEP is running at the PC. (If the remote never transmits "junk" when powered on or off, or when you disconnect and reconnect the cabling, then you'll have no problems. You can still turn on or off, or disconnect or reconnect.)

It is appropriate to use /NOTEST while you're experimenting with PEP, or if you're having some problems with a particular remote application, or while you're in the process of testing your cabling. Or, if the DSR pin on your asynchronous adapter is "sticking," as explained in Section 3.7.

NOTE 1: Once you've loaded PEP, you may remove the PEP diskette from your drive. However, the diskette which you have in the "boot drive" should contain the DOS system file COMMAND.COM since the operating system may need this after you terminate PEP. (See SET and COMSPEC in the DOS manual for pertinent information.)

NOTE 2: If you want to use a small communications buffer, or specify that the remote is sampling DSR, or use the /NOTEST option, then this MUST be done by providing an option as part of the command line. None of this can be done once PEP is running.

NOTE 3: You may use capital or small letters, or any combination of these, in typing the command options.

NOTE 4: If PEP.COM can't find PEP.OV, you'll encounter the message

ERROR : Unable to locate PEP.OV

See Chapter 5 for more details.

NOTE 5: If you're using a hard disk, then it's probable that your operating diskette has subdirectories. You'll find that allowing PEP.OV to be in the root directory of the default disk is particularly advantageous. Without having multiple copies of PEP.OV on it, you can invoke PEP from any directory of your default disk.

To minimize difficulties, PEP.COM should be in a directory which is searched by the command processor when it attempts to load and execute a program. In other words, the directory which contains PEP.COM should appear in the PATH list of directories. (The PATH command is usually contained within the automatically executed batch file AUTOEXEC.BAT.) So, if you haven't already done so, place a copy of PEP.COM in one of these directories, and erase the redundant copy in your root directory.

Henceforth, entering the command PEP from anywhere on the default disk will cause PEP to be loaded and executed.

If you're not familiar with the PATH command, consult your DOS manual.

NOTE 6: PEP will require that your remote computer transmit its data in the following bit format: a "start bit," followed by exactly eight bits (including both data and parity), followed by one or two stop bits. Your remote will automatically generate the start bit, and will generate the stop bits as you instruct.

Regarding the eight (repeat: EIGHT) bits which must be "sandwiched" between the start bit and the stop bit(s):

If you're transmitting 7 bit data from the remote, you MUST also transmit a parity bit. (The parity may be even or odd, or any other kind, so long as a parity bit is actually transmitted with 7 bit data. Note that for most computers and most software, "no parity" or a clause like "PA = N" inhibits the transmission of a parity bit; this would of course be INAPPROPRIATE if you're transmitting 7 bit data.)

If you're transmitting 8 bit data from the remote, you MUST NOT transmit a parity bit. (So, you probably need to suppress parity, using "no parity" or a clause like "PA = N.")

The information in the preceding two paragraphs is EXTREMELY IMPORTANT. Please try to keep this information in mind, especially as you begin using PEP to receive data, as explained in Chapter 3.

NOTE 7: Forgive just one more technicality, if you will. It's aimed at a very small audience.

PEP is designed to operate in an asynchronous communications environment, using the standard asynchronous hardware within your PC, as opposed to a synchronous environment. This will probably be of no concern to you if you're using PEP to capture data transmitted by a microcomputer, since synchronous communications hardware is both unusual and very expensive for microcomputers. However, if you're using PEP with a large computer network, a minicomputer, or a mainframe, please take note: you'll need to transmit data to your PC asynchronously.

Chapter 2.

Telling PEP What It Needs, and What You Want

PEP operates in two phases.

In Phase I PEP presents you with a questionnaire, which will provide the information required to establish a useful linkage between your computers.

In Phase II, the transmission phase, PEP will receive the data which is "printed" by your remote computer to the PC, and process it according to the instructions you've given in Phase I.

In spite of its simple title, this chapter will acquaint you with a wide variety of information about PEP. You'll learn in detail how PEP works, what PEP does during the transmission phase, what types of data conversion PEP provides, and how to complete PEP's questionnaire.

2.1 Handling the incoming data

As data enters your PC from the remote computer, it is assembled into a sequence of characters. During Phase II these characters may be displayed on your screen, saved to a diskette, or printed on your printer. You may elect to have PEP do any combination of these three things, including all three of them simultaneously.

How PEP works

When you've completed the questionnaire, PEP will enter Phase II of its operation.

Initially, PEP will wait for incoming data. While waiting it will continuously monitor the communications lines.

Once incoming data is detected, PEP will make sure that the communications lines are in their proper state and that the incoming data is correctly "formed." (Data bits, stop bits, and the like.) PEP will continue to monitor what's happening for another twentieth of a second, as further data enters the PC.

If everything continues to be in order, PEP will assume that the data thus far received is valid. If you've elected to save the incoming data to diskette, PEP will then (and only then) open a diskette file in which the incoming data will be stored.

PEP will process the data thus far received, and will continue to process all of the subsequently received data until a "terminating event" occurs.

By "processing the data" we mean displaying, saving, and/or printing it. A "terminating event" occurs when either

(a) a five-second pause is detected in the transmission, or

(b) a fatal transmission irregularity is detected, or

(c) you manually terminate or restart PEP, or

(d) the operating system is unable to write on your diskette, or print on your printer, and produces a system error in trying to do so

When a terminating event occurs, PEP will immediately stop processing the incoming data. This implies closing the diskette file which is currently receiving data, if there is one.

If condition (a) arises then PEP will beep to indicate that you may commence the next transmission, and will print an appropriate message on the screen. PEP will again be idling, waiting for further incoming data.

At the very beginning of each transmission, PEP carefully studies the RS232 data and control lines both before and briefly after receiving the first character. It is in this way that PEP is able to differentiate between the start of a valid transmission and a line glitch caused by a disconnect between transmissions. Should PEP detect a line disturbance (a disconnect) between transmissions, then it will wait until the lines are normalized and stable for at least five seconds before returning to operation.

HOWEVER — if you're running PEP with the /NOTEST option, then PEP will not monitor the RS232 control lines, and will not be able to detect a disconnect of the remote computer.

Displaying the data

While PEP is running, you'll see two bold bars running horizontally across the screen.

If during Phase I you've told PEP to display the incoming data, it will do so between the horizontal bars. If PEP wasn't requested to display the incoming data, it will instead display a running total of the number of characters received in the current transmission.

If you're transmitting "binary" data, such as the unreadable object code of a compiled program, you should NOT display it. It will appear nonsensical, but as well, some of the incoming characters will cause the PC to beep. (These are the "bell characters" with ASCII number 7.) These beeps will prevent you from conveniently knowing when PEP is ready for the next transmission, since they will sound exactly like the beep that PEP makes between data packets.

Saving data to diskette

If you've told PEP you want the incoming characters saved to diskette, PEP will open a diskette file for this purpose. This occurs only after PEP has received the first character of the transmission, and has continued to receive characters for another twentieth of a second without any problems.

The first diskette file which PEP uses is the file

REMOTE.001

on one of your drives (the one you've requested in Phase I).

PEP will save all of the received data in this file, until a terminating event occurs. At this point, PEP will close the file which has been receiving the incoming data.

If the file has been closed as the result of a pause in transmission, then PEP will beep and wait for the next transmission. The next transmission will be stored in the file REMOTE.002, the following transmission will be stored in REMOTE.003, and so forth.

Printing the data

If you requested PEP to print the incoming data, it will oblige. The printout will always be to the system device PRN: (also known to the DOS as LPT1:).

Unless you're using a print spooler, then as far as PEP is concerned the printout will probably proceed far more slowly than the reception of data into your PC. Ordinarily PEP uses a 32,768 byte input buffer for the data it is receiving, and each received character will be placed in a single byte of this buffer. (A "buffer" is a portion of computer memory which has been set aside for a particular purpose, usually the temporary storage of data. A "byte" is a location in the computer which is adequate for the storage of a single character.)

Because of the buffer, your transmissions will be able to "get ahead of" the printout by about 32000 characters. (32,718 to be exact.) PEP will assert control over one of the communications lines when the buffer is nearly full, preventing the remote computer from sending data for a while — until there is room in the buffer for 300 characters. Characters are "emptied out" of the buffer as they are printed.

So, unless you're sending your PC an enormous file, the remote computer will be able to transmit the data as fast as it can to the PC, without stopping. The PC will gradually catch up as this data is printed, and while this is happening you'll be able to do something else with the remote computer. Just as if your PC is a printer with a "print spooler."

But . . . if you're saving the incoming data as well as printing it, then you shouldn't ask your remote computer to send another file to the PC until the printout has been completed and the PC has beeped. PEP will only start looking for a five-second pause between transmissions AFTER IT HAS EMPTIED THE BUFFER. More on this in Chapter 4.

2.2 Special keystrokes

In the next section, we'll consider PEP's questionnaire. Before we do, you should know about some special keystrokes which allow you to supply a "default" value, restart PEP to correct a mistake or change PEP's operation, or terminate the program.

You may terminate, pause, or restart PEP by pressing a single keystroke combination. These are listed on the horizontal bar at the bottom of the screen : Ctrl-H = Hold, Ctrl-R = Restart, Ctrl-T = Terminate.

Ctrl-T will end PEP's operation, and return you to the DOS.

You'll use Ctrl-R to modify the information you've supplied to PEP, either after you realize you've made a mistake, or after you've received some data and you wish to receive further data under different circumstances. When you press Ctrl-R the screen will clear, any diskette data files you're using will be closed, and if you're printing then the printout will immediately stop. PEP's initial screen, the one with the copyright notice, will then be displayed as if you've just loaded PEP.

However, there are two crucial differences between loading PEP and restarting PEP. First, the numerical suffix (.001, .002, .003 and so on) which is used with the diskette files starts with .001 when you load PEP. However, when you restart PEP the sequence will continue in its natural order, where PEP left off. This will help you avoid overwriting an important data file.

Second, while you're completing PEP's questionnaire, it will be easy to supply PEP with some previously selected "default values." Initially, as you loaded PEP, these default values are as described in the remainder of this chapter. After you restart PEP, the default values will become those which you last supplied, before you pressed Ctrl-R.

Ctrl-H will be useful when you're actually receiving data, and you wish to pause or "hold" for a moment to read the incoming data on the screen. In Phase I, while you're providing PEP with the information it requires, this keystroke will have no effect.

During Phase I, the space bar and the enter key have a special significance. Pressing either key will cause PEP to accept the default value of a parameter, or to accept a list of options which you've just edited.

2.3 Phase I. Step 1

Whew !

Finally, we're in a position to examine and respond to PEP's questionnaire.

Load PEP as explained in Chapter 1. PEP's initial screen, with the copyright notice, will appear on the monitor.

IMPORTANT NOTE: In the remainder of this chapter, we'll be describing some "default values" which PEP assumes as it is loaded. These values may change if you elect to restart PEP. Each default value will then become the value you last supplied, before you pressed Ctrl-R.

Between the two bold horizontal bars you'll see a title,

STEP 1. Transmission protocols :

In step 1, PEP will request you to supply five pieces of information regarding the way in which your computers will be communicating.

You'll first be asked to identify the communications port on the remote computer. (This is only for your later reference, PEP doesn't actually use it.) Respond by pressing one of the number keys 1 through 4, or by pressing the space bar or the enter key (either will provide the default value, 1).

(Most of this explanatory information is on the screen, so you'll probably never need to refer to this part of the documentation again.)

Second, you'll identify which asynchronous communications adapter you'll be using on the PC. "Asynchronous communications adapter" is the technical name for your PC's standard RS232 card, or serial card, as it's otherwise known. The DOS recognizes two such cards, numbered 1 and 2. Respond by pressing a 1 or a 2 to indicate which adapter is connected to the remote computer. You may also press the space bar or the enter key to supply the default value, 1.

Next, you'll indicate the speed at which data will be sent to the PC. The available options are 300, 600, 1200, 2400, 4800, and 9600 baud (bits per second). Indicate your choice by

pressing the number key which is the first digit of the speed. Pressing the space bar or the enter key will provide the default value, a 9 for 9600 baud. This is the fastest transmission rate that the PC's asynchronous adapter can handle, and it allows data transfer at up to 960 characters per second.

You might have good reason to believe that 9600 baud won't work — for example, your cables are more than 50 feet long, or the remote computer doesn't transmit at 9600 baud. We suggest that you at least try 9600 baud as the transmission speed, if you think that this might conceivably work. PEP can easily handle data which is being received at this rate.

Next, tell PEP how many data bits are being used, 7 or 8. If you're not sure, then ordinarily you should try 7. The space bar and the enter key will each supply the default value, a 7.

Finally, indicate the number of stop bits being used. This is usually 1. Pressing the space bar, the enter key, or 1 will indicate one stop bit (the default value) and pressing 2 will indicate two stop bits.

2.4 More Phase I responses. Step 2

At the end of Step 1, PEP will replace the copyright notice at the top of the screen by an abbreviated summary of what you've specified so far. This summary will be updated at the end of Step 3, and will remain visible while you're using PEP to receive data.

Between the horizontal bars, you'll see

STEP 2. Handling the incoming data :

and a list of options will appear just below. These are D = display, S = save, P = print, and L = long pause (20 seconds). Enter a list of the options to be used in handling the incoming data.

After you're familiar with PEP, we suspect that you'll usually want to both display and save the incoming data. For this reason you'll initially be presented with the default options list

DS

so that you can just press enter or the space bar. As explained earlier, "displaying the data" means printing it on the screen as it is received, and "saving the data" means saving it to a diskette file.

Changing the list is easy. To "toggle" one of the options, just press the letter key corresponding to the option. This will change the list, by removing the letter if it already appears, or adding it if it does not. Please experiment to see how this works, it's an unusually simple editing scheme. Leave just the letter D visible.

After you've edited the options list, you may press either the enter key OR THE SPACE BAR to accept the list as it stands.

The letter P should appear in the options list if you want to print the incoming data. For now, don't request a printout — in other words, don't press enter (or the space bar) when a P is visible in the list.

If an S is entered as part of the options list, then PEP will ask you to identify (by pressing a single letter) the drive on which you want to store PEP's files. If you press the space bar or the enter key, the files will be stored on the "default drive." (The default drive is the drive whose letter code was visible as part of the DOS prompt when you loaded PEP. Most likely, drive A.) You may certainly specify a virtual disk, if one is installed in your system, or a drive "alias" created by DOS's SUBST command (DOS 3.1 or higher).

How about L?

Some applications, and some remote computers, are very slow. If there are five-second pauses in the printed output of the program running on the remote, then PEP will have problems. The option L is intended to compensate for this.

Remember that PEP normally interprets a five-second pause as a "marker," to indicate the end of one data packet and the beginning of another. If you're saving the incoming data to diskette, PEP will close the data file which was recording the data prior to the marker, and open a data file to record the data following the marker. So, a slow application at the remote may generate a dozen or more files from a single printout, if a pause of only five seconds is interpreted as an inter-packet marker.

If you use the option L, a pause of 20 seconds will be used to mark the end of one transmission and the beginning of another. You'll still encounter some problems with really sluggish programs (not terribly serious ones — see Section 4.2) but even a horribly slow computer's attempt to copy a diskette (or cassette) file to an RS232 port should cause no problems with a pause of this length.

So much for an explanation of the options available in Step 2. For the time being, please enter

D

(that's right, only D) as your response to Step 2. Don't press the enter key when P is visible, and please don't press enter (or the space bar) when S is visible either. Just using D will allow you to watch the incoming data, although it won't yet be saved or printed.

2.5 Phase I. Step 3

Between the horizontal bars, you'll see

STEP 3. Data interpretation (conversion) mode :

In step 3, you'll press a letter key to indicate how you want the incoming data to be "interpreted" or "converted."

This conversion is principally a concession to the different ways in which data is treated by different computer operating systems, and to the ways in which data is transmitted via the RS232 interface in your remote computer. Most printers incorporate internal dip switches to accommodate to these differences.

The PC-DOS and MS-DOS operating systems require that "text files" (readable files, in the form normally printed or transmitted) explicitly separate lines of text using two characters. These are a carriage return, ASCII character number 13, and a linefeed, ASCII character number 10. Both of these characters should appear, in this order, between lines of text on your PC.

(By the way, some MS-DOS applications are flexible enough to handle and, more or less, properly separate lines of text which aren't quite correctly marked.)

Some operating systems, or data transmission protocols, use only a carriage return between lines of text. And some applications use both a carriage return and a linefeed, but occasionally in nonstandard order (linefeed first), possibly due to a programmer's oversight.

If you believe that the incoming data should be accepted verbatim, without any changes whatsoever, press S. S stands for "straight through." Keep in mind that you can always perform a conversion after the data is on a PC diskette, using a fairly straightforward BASIC program for example.

If you have reason to believe that only carriage returns will be used to separate lines of transmitted text, press L. L stands for "add linefeeds," meaning that a linefeed character will automatically be appended to the incoming text immediately after each and every carriage return character.

If you're transmitting binary (unreadable) data, or the incoming data will have elaborate "escape sequences," or you're not sure whether to use S or L, you should use S.

2.6 The special conversion option X

If you intend to receive and save a file which is being sent by a problematic application, you might try using the conversion option X. This will put linefeeds and carriage returns in the appropriate places for MS-DOS, in the appropriate order, and will replace any formfeed characters (ASCII 12) by a backslash character (\).

We STRONGLY RECOMMEND that if you're having early problems using PEP, you try the option X.

The option X will make certain types of underlining visible. This is a rather restrictive class of underlining, produced by a remote application executing a carriage return and then printing the underline.

Option X performs its conversion very simply. The conversion ignores all incoming linefeeds, and converts carriage returns to carriage return - linefeed sequences. Every formfeed character is replaced by a backslash.

We mention these details primarily to motivate the following advice : DON'T send fancy escape sequences to the PC using a remote application, if you're converting the incoming data with X. Why? Each and every ASCII 10, 12, and 13 of the incoming character stream will be modified by this conversion, whether such characters are intended for line demarcation ("carriage control") or are intended to be used as part of an escape sequence.

NOTE that this special conversion is designed to place a problematic text file conveniently in MS-DOS format on your PC diskette, principally for further reference or editing. The backslash is used rather than the standard formfeed character to allow you to edit the PC file easily, regardless of how simple your text editor is.

You should NOT use the data handling option P ("print") in conjunction with the conversion option X, since the printer would not receive the correct formfeed character to make the printout acceptable.

PEP FOR THE TI-99/4A. We would be remiss if we didn't accommodate for some of the peculiarities of COMPANION, Intelpro's word processor for the TI-99/4A. The conversion option X will handle this situation; see Appendix D for further details.

2.7 End of Phase I

Once you've pressed a single key to respond to Step 3, PEP will produce two summaries of the information you've provided. Between the horizontal bars, you'll see an informal verbal summary of the general way that the incoming data will be handled. The abbreviated technical summary above the upper horizontal bar will be updated. You'll see the prompt

PRESS THE SPACE BAR OR ENTER
TO BEGIN DATA RECEPTION

between the horizontal bars. Go ahead, press either key.

You're now in Phase II. Don't worry about what you see on the screen — press Ctrl-R to restart PEP. To review what you've learned in this chapter, run through the questionnaire a

few times, pressing Ctrl-R each time you reach Phase II. Observe that the default values which PEP supplies will change each time you restart the program; they assume the value which you used previously.

Finally, press Ctrl-T to terminate this run of PEP, AFTER you've read the remainder of this chapter.

2.8 A review: the keyboard

PEP has been designed so that the keyboard has a simple, consistent, and convenient meaning. Let's review.

As noted in Section 2.2, Ctrl-R and Ctrl-T may always be used to restart or terminate PEP. During Phase II, Ctrl-H may be used to "hold" PEP, pausing both the processing of data and its transmission.

Both the space bar and the enter key may be consistently used in Phase I to provide a particular value for a parameter, or to input an edited option list.

Pressing either key when you first load PEP will provide a "default value," a parameter value which in our opinion is most likely to be useful.

Pressing either key after you restart PEP will result in the confirmation of a value previously specified. This enables you to move quickly through the parameters with the space bar or the enter key, pausing only to change those parameters receiving special attention.

2.9 The screen

PEP's screen, although sometimes quite cluttered, will consistently provide a summary of any detail you may require.

Remember that it is NOT the spirit of PEP to require any attention at the PC while you transmit data. On the contrary! The information on the PC's screen is intended only to allow quick diagnosis of a problem, or to provide confidence in the status of a transmission.

The horizontal bars

On PEP's screen, the bottom horizontal bar usually displays a reminder regarding the use of Ctrl-H, Ctrl-R, and Ctrl-T. When you've pressed Ctrl-H to pause PEP, a message to this effect will appear on the bottom bar. When you press another key, PEP will resume normal operation, and this message will be replaced by the usual message about the control keystrokes.

If you see a backslash (\) on the lower bar, this indicates that the program module PEP.OV has been located in the root directory rather than in the current directory.

The upper bar indicates the size of the input buffer being used by PEP. During Phase I you'll also see the title "Phase I. The questionnaire" on this bar. During Phase II the bar will display either a title, "Phase II. Data reception," or else (if you're saving the incoming data) the name of a diskette file.

If you're saving incoming data to diskette, the upper bar will indicate "Incoming File = REMOTE.XXX" when the file REMOTE.XXX has been opened and some incoming data has actually been written to it. Between transmissions, while PEP is waiting for valid incoming data, you'll see "Next file = REMOTE.XXX" indicating that the file REMOTE.XXX is not yet open and has not yet stored any incoming data.

Above the upper bar

When you load PEP, the copyright notice appears above the upper bar. As you progress through Phase I, most of this area will be used to summarize the communications parameters which you supply. At the end of Phase I, a full summary of the parameters will appear here, and will remain until you restart or terminate the program.

The parameters will be summarized in the format

Cards: PC = 1 Remote = 2 Mode:L
Protocol:BA = 300.DA = 8.PA = N.ST = 2.LF Options:S

The upper left entry (Cards) will indicate the asynchronous adapter being used at the PC (adapter number 1 in this instance), and the RS232 port being used at the remote computer (here, port number 2). Below this, the communications protocols in effect will be listed, with a period being used as a separator.

In the current example, communications should proceed at a rate of 300 baud, with 8 data bits and 2 stop bits. As a reminder that some computers may require the suppression of an extra parity bit when sending 8 bit data, you'll see the expression PA = N. This abbreviates "parity = none," and will appear only as a reminder, and only when you've specified 8 bit data.

If you're using the conversion option L (add linefeeds), the expression LF will appear in the protocol to remind you to send data with suppressed linefeeds.

The other abbreviations are fairly obvious. BA abbreviates "baud rate," DA abbreviates "data bits," and ST abbreviates "stop bits."

The upper right entry indicates the conversion mode being used. This will be either S (straight through), L (add linefeeds), or X (special conversion).

Below this, the data handling option list is reproduced. D still means "display," S means "save," P means "print," and L means "long pause."

The special command line options (if any) you're using will appear in the upper right hand corner of the screen. This space will display the message

/NOTEST /D

(or a part of it) if you're using the /NOTEST or /D option.

Between the bars

Between the horizontal bars is a scrollable window, used to display incoming data and to provide various messages. Prompts to transmit the next data packet, warnings about the loss of the communications channel, and all error messages arising in Phase II will appear here.

This window is used during Phase I to display the three-part questionnaire.

Redrawing the screen

The DOS's "Critical Error Handler" is notorious for ruining a screen layout. To redraw the screen, restart PEP with Ctrl-R, preferably between transmissions.

Chapter 3.

Receiving Some Data Files

We'll assume that you've read the manual up to this point, that you're comfortable with PEP's questionnaire, and that you have a cable to connect your PC to the remote computer.

(Again, the cabling is explained in Appendices A, B, and C. You should read these VERY CAREFULLY before trying to obtain a cable.)

We'll also assume that the cable securely connects the two computers, and that you know which "ports" or "adapters" are connected.

In this chapter, we'll put PEP to work by receiving data files from the remote computer, and displaying these on your PC's screen.

3.1 The first transmission

Let's proceed slowly and carefully at first, to make sure that you have the right kind of cable attaching your two computers, and to give you some confidence in PEP.

Turn on both computers. It probably won't matter which computer you power on first, your PC or the remote. In our experience, most PC's are rugged, very unlikely to be perturbed by any normal incoming line surge generated along your cable, and also very unlikely to create any problems with the remote.

Just let your remote computer idle for now — at the very least, don't ask it to send any data yet.

Get PEP running, and complete the questionnaire as best you can, supposing that you'll run a very simple program in a moment to send some data to your PC. (Just handle the incoming data with D, to display it.) When you're finished with Phase I (the questionnaire), press the space bar or the enter key to commence Phase II.

Between the two horizontal bars, PEP will display a single solid rectangle followed by the message

Ready to receive next incoming file

and we do hope that that's all you see between the bars.

If not, you'll see an "error" message (two solid rectangles, followed by the message). This is caused by an improper condition at the Data Set Ready pin of your PC (it will test "off"). Either

(a) you've specified the wrong adapter for the PC

(b) your cable is loose

(c) there's a problem with the way that your cable is wired

(d) the remote computer is not powered on, or the port to which the cable is attached isn't powered on

(e) there's a problem with the RS232 hardware on the remote computer

(f) there's a problem with the RS232 hardware on the PC

(g) all is well, but the remote computer is not asserting the required control over its RS232 pins.

That's a lot of potential problems!

3.2 Problems?

The most likely problem is (g).

Unfortunately, we suspect that the second most likely problem is (c). Don't be upset, either at yourself or the technician who wired your cable, if this finally proves to be the problem. Few people understand the sketchy documentation that computer companies provide.

The quickest solution to the present problem is to run PEP with the /NOTEST option. Then PEP will not test the PC's DSR pin. However, it is certainly preferable to avoid using the /NOTEST option, and to correctly diagnose the problem, if you want PEP to be sensitive to disconnections of the remote.

Problems (a), (b), and (d) are quite easily solved, and I suggest that you look at these first.

For problem (a) : If you have two asynchronous adapters on your PC, rather than moving the cabling you should press Ctrl-R to restart PEP. This time, indicate that you'll be using the other adapter.

Once you've checked (a), (b), and (d), you should consider the possibility that there isn't any real problem, in other words, you're in situation (g). Run PEP using /NOTEST, or try running different software at the remote.

On a reasonably sophisticated micro, like an Apple II, a problem of type (g) could arise if you've run a program which modifies the communications environment but fails to reestablish the original environment before terminating. Unfortunately, this will make problems of type (g) appear to be intermittent, since the status of your communications hardware will depend upon the history of the computer since you turned it on.

You can tackle (e) and (f) by using your RS232 hardware to do something else, like printing to a standard printer (one having an RS232 interface) or interfacing to a modem. If you get good results, this suggests that the problem isn't in the piece of hardware you're testing.

3.3 No problems?

Great.

Now, decide upon a piece of software which you'll be running on the remote computer to test the communications linkage. You should be able to control, or at least know, the baud rate, number of data bits, and number of stop bits which the software will be using to transmit data, as well as the number of the RS232 port which it will transmit through. The port should of course be the one which is connected to the PC.

The software could be a very small BASIC program which you'll write for yourself. In fact, this is probably the very best way to begin, if you know how to program and have a good idea how to control RS232 communications.

Run your program, and print through the RS232 using the baud rate, number of data bits, and number of stop bits which PEP is expecting to receive. Adjust PEP, by restarting it, if you have to. (If you're using 8 bit transmission, you may find it necessary to suppress the transmission of a parity bit, using a clause like "PA = N" at the remote computer.)

You should be able to muddle through. If not, the "Transmitted Data" pin of your remote is probably not wired to the "Received Data" pin of the PC's asynchronous adapter. In other words, you have a minor problem with your cable. If all else fails, run PEP with the /NOTEST option.

3.4 Error messages

For now, don't worry if you're encountering some error messages from time to time.

If your cabling is correct and your hardware is behaving as it should, then these error messages are "line errors" resulting from an incorrect baud rate or improper "framing" due to the wrong number of data bits or stop bits. When such a message appears (the words "framing error" or "break detect" are the key), then in response to the error you can press Ctrl-I to request PEP to ignore all of the line errors detected during the transmission of the current data packet. You might receive junk, but at least this will get PEP moving.

Ctrl-I is intended primarily to get you through the first difficulties you encounter when using PEP. Although this may permit you to receive the data which you're trying to get, pressing Ctrl-I every time you have a problem is a poor substitute for correctly diagnosing the problem.

We'll discuss error messages again in chapter 5.

3.5 A check of your cabling

Sometimes, PEP will try to prevent your remote computer from sending further data for a while. It will do this by controlling one of the communications lines.

For example, if you press Ctrl-H during data transmission, this will "hold" PEP until you press another key. Additionally, it should prevent the remote from transmitting during the pause.

(We recommend that you use Ctrl-H rather than the system's pause key, Ctrl-Num Lock, since Ctrl-Num Lock will freeze your PC but not the remote. By the way, the remote isn't "frozen," rather it will refrain from attempting to transmit until the communications lines return to their normal state.)

To make sure that Ctrl-H works properly, we suggest the following:

While PEP is idling between the reception of data packets, press Ctrl-H. Leave PEP in its paused state.

On your remote computer, run an application which will output a small amount of data to an RS232 port. The application should "stall," or else it may display a "timeout" error message. In either case, your cabling passes this test. Just to be on the safe side, press a key on your PC and then run the application again. The application should now run normally, sending the data without stalling or timing out.

If your remote didn't stall or timeout while the PC was paused, either you have a problem with your cable or you should try using PEP with the option /D (or without /D, if you're already using /D). Or, you have a hardware problem with the PC's or the remote's RS232 circuitry.

See Note 1 for a more rigorous test.

3.6 More about Ctrl-H

As we've seen, Ctrl-H is useful in testing the cabling between your computers.

When PEP is in Phase II, pressing Ctrl-H will cause the solid bar at the bottom of the screen to display the message

HOLD. Press any key to resume.

Once you've pressed another key, this message will be replaced by the usual reminder about the special keystrokes, and the transmission will recommence.

Ctrl-H allows you to pause PEP for a moment if you've wandered over to your PC and you'd like to read some of the incoming data on its screen, or if you need to adjust some PC hardware (like a printer), or if the remote computer requires some adjustment. PEP's "clock," which measures the time elapsed between transmissions, is frozen by Ctrl-H, so as far as PEP is concerned, this pause may be of any duration whatsoever. In other words, you needn't worry that Ctrl-H will cause PEP to think that further data from the present data packet is from another data packet.

We hope that your remote computer will permit you to use this feature.

If at first there seems to be a problem, it may be that your remote is "timing out" when it perceives that the "printer" (your PC) is unavailable for some time. If the remote is smart enough to do this, then usually its operating system allows easy adjustment of the timeout interval. Try to adjust it so that it never times out, or so that it times out only after several minutes. Then using Ctrl-H should be no problem, no matter what application you're running on the remote.

If you're printing the incoming data, Ctrl-H will be a bit sluggish. That's because sometimes the printer will keep the PC busy for a few seconds.

If you can't adjust your remote so that it avoids timing out altogether (or for, say, a minute), then we suggest that you not use PEP to print a large incoming data packet (more than 32,000 characters). PEP will prevent further transmission if its 32K input buffer is almost full (only 50 spaces left), and will only reenable transmission when this buffer has room for 300 characters. Since the buffer is emptied 255 characters at a time, you should avoid timing out in the period of time it takes to print 255 characters, including performing all of the carriage returns, linefeeds, and formfeeds.

3.7 A final test of the hardware

Run PEP, without using the /NOTEST option.

While PEP is idling between transmissions, turn your remote computer (INCLUDING the RS232 circuitry) on and off a few times. Each time, wait about ten seconds between turning it on and off.

For many computers, turning "on" means turning the power on and then opening a communications channel (via the DOS, or using OPEN in a BASIC program). Turning "off" means closing the channel, via a CLOSE in BASIC, for example.

Assuming you're running PEP without the /NOTEST option, PEP should indicate that the communications lines are available about five seconds after the remote is turned on, and that they're not available immediately after it's turned off. If this isn't happening, and your cable has performed well, then you probably have a hardware "problem" at your PC. The DSR pin on your asynchronous adapter is "sticking," not responding to changes in the communications lines.

This "problem" isn't really a problem at all, just an unfortunate peculiarity of your asynchronous adapter. If your PC is equipped with two adapters, try the other one, and use it if its DSR input provides evidence of disconnections of the remote. See Note 2.

3.8 Experiment ! ! !

I do hope that your experience thus far has not been too harrowing, frustrating, or exasperating.

However difficult it has been to get the right cable, and to get your remote computer working well, you're now in a position to put PEP to work.

Please experiment, give PEP a real workout! Just display the transmitted texts for now.

NOTE 1: As a more rigorous test of your cable, and the functioning of Ctrl-H, we suggest the following:

On your remote computer, write a short BASIC program which prints the sequence 1, 2, 3, . . . both on your remote's screen (or LCD display, or printer) and also through the RS232.

Run the program. It should be producing the number sequence locally, and the sequence should be appearing on the PC's screen as well. Press Ctrl-H and then see what the remote is doing. If it has paused, your cabling is perfectly correct. If it has produced an error message on the remote, your cabling is correct but you may have a few problems making use of this feature of PEP with applications like the one you're running on the remote. Solvable problems, we think, having to do with "timing out."

If the remote just keeps on producing numbers, you have a problem of the kind mentioned in 3.5.

NOTE 2: On the authoring machine, an IBM PC-XT, one of its asynchronous adapters has a DSR pin which "sticks," and the other adapter's DSR pin (on an AST SixPack-Plus) doesn't stick. For this reason we run PEP through the AST's adapter.

If the DSR pin on all of your asynchronous adapters "stick," then you should run PEP using the /NOTEST option. Unfortunately your PC will not be able to detect disconnects, and you should avoid these while PEP is running.

The current version of the RS232 standard, RS232C, indicates that for the chip inputs to be valid, they should be driven by a remote device. So, a sticky DSR is not technically in conflict with RS232, since when the remote is turned off the DSR pin is no longer driven.

For the standard IBM Asynchronous Adapter, the Technical Reference Manual tells us that voltage levels at the communications pins determine whether they should test on or off. For voltages between -3 volts and -15 volts, a pin should test "off," and for voltages between 3 and 15 volts a pin should test "on." Other voltage levels are "invalid levels," and the reaction to these at the adapter is not specified. Since a disconnect will result in a voltage at or very near zero, the adapter is not technically required to respond to this change in the communications environment.

Chapter 4.

Putting PEP to Work

Your worries are over. Your cable works, and you've discovered that you can reliably transmit data from your remote computer to your PC.

In this chapter, you'll learn how to get the most out of PEP. In addition to just displaying data on your PC's screen, you'll be able to save the data to diskette and print it on the PC's printer. You'll see how to use the PC and its printer as a print spooler for your remote computer. You'll learn when is the best time to turn the remote computer off or on.

And you'll learn how to transmit a collection of files, automatically, to the PC.

4.1 Saving the incoming data

You probably purchased PEP to help you to "copy" data to your PC's diskettes from another computer.

So, more than anything else, you'd like PEP to save the incoming data on one of the PC's diskettes. To do this, enter a "data handling options list" which includes the letter S. This is done in Phase I, which may be invoked at any time by pressing Ctrl-R.

Once you've entered such an options list, PEP will ask you to specify the drive used to save the incoming data. Press any letter key to specify this drive, or the space bar or enter for the default drive. You may use a virtual drive if one is installed in your system, or a drive "alias" if you're using DOS 3.1 or higher.

In Phase II, PEP will store the incoming data to this drive, in the drive's current directory, giving the first data packet the diskette name REMOTE.001. The next transmission will be stored in REMOTE.002, the subsequent one in REMOTE.003, and so forth. PEP will not attempt to store more than 999 files, and will automatically terminate after it has received the file REMOTE.999.

These file names have been selected for your convenience, keeping in mind that it's very easy to rename or move these data files after you've terminated PEP. (You'll use the DOS commands RENAME and COPY to do this.)

It is ESSENTIAL that you either rename or move these files before you run PEP again and target the same diskette to receive files, so that you don't inadvertently overwrite them when PEP starts to save new data.

You may safely press Ctrl-R to restart PEP, in the sense that PEP will never overwrite files created before the restart. PEP will keep track of what's happened since you started the program, and will save further incoming files with suffixes larger than those already used.

4.2 Saving to separate files

Remember that one of PEP's strong points is that it allows you to remain seated at the remote computer and send a sequence of files to your PC, and these will be recognized and stored as separate files on the PC. For instance, you can send 31 distinct data files from the remote computer to the PC, and they'll be stored as 31 distinct files on the PC, all without leaving the remote computer.

The key is the pause between transmissions. Ordinarily PEP recognizes an end of transmission when it has finished processing the data it's received and then observes five seconds of silence on the transmission lines. (Since PEP is extremely fast, this usually takes barely over five seconds, unless you're printing the incoming data.) You can lengthen 5 seconds to 20 seconds if you select the L data handling option; this would be appropriate if your remote is running a slow-printing application.

After the end-of-transmission pause, PEP will close the file being used to store the previous transmission. Then it will beep, indicating that you may proceed to transmit the next data file, and that it will be stored on the PC as a file distinct from the previous one.

Whatever happens, PEP will save everything validly transmitted through the RS232 lines, period. (Unless the operating system refuses, since your PC diskette is full, for example.) If you've forgotten to wait for the beep before you begin your next transmission, or if the transmitting software has paused for a bit too long, the data will still be written to the PC's diskette. There will be one file where you intended two in the first instance, and two files where you wanted one in the second instance.

A few minutes with a good word processor will cure either problem if you're dealing with text files whose contents you're familiar with.

If you're transmitting binary (unreadable) files, then using the DOS COPY command so that it will glue together separate files will cure the second problem. On the other hand, it's a bit difficult to split binary files; retransmit them if you've incurred a problem of the first type.

4.3 Keeping track of the file name

While incoming data is being written to the PC diskette, the upper horizontal bar on the screen will indicate the name of the file which is receiving the data. Between transmissions, the bar will show the name of the next file to be used.

4.4 Printing

To print the data entering your PC, include P in the data handling options list. This will either print a verbatim copy of the incoming data, if you use the conversion option S, or a slightly modified version of the data with another conversion option. You can of course also include D and/or S in the options list, to display and/or save the incoming data as well.

4.5 A print spooler

You can use your PC together with PEP as a print spooler, if you use only P as your data handling options list. The incoming data will not be displayed or saved, but a count of the incoming characters will appear on the screen as each block of characters is printed. (Incidentally, these blocks will probably appear to be rather large, about 255 characters. This is because as PEP's input buffer fills, it is emptied in fairly large chunks, with 255 characters being the maximum chunk size.)

The remote computer will be able to send characters as fast as it can, while PEP and the printer do their best to catch up. Only in the case that you transmit so much data to your PC that PEP falls behind by more than 32000 characters, will PEP slow output from the remote computer.

If you've been reading this manual carefully, you might now have an objection. "What about the beep between data transfers. If I have to wait until the printer catches up and PEP beeps before I print another file, then that's not much of a print spooler!"

Agreed, and fortunately YOU DON'T HAVE TO WAIT! The pause between transmissions is recommended only so that PEP can place distinct transmissions into distinct diskette files, so if you're not saving the incoming data to diskette, you don't need to provide a pause between transmissions. In other words, PEP is a great print spooler, if 32K (16 pages) of "spool ahead" is adequate for your purposes. Just ignore the beeps!

4.6 When can I disconnect?

If you're running PEP with the /NOTEST option, then you should not disconnect or turn your remote computer on or off while PEP is running.

Why? Well, the disconnection may cause a line surge along the RS232 cable which will appear to your PC's hardware to be an attempt to transmit a character. Since /NOTEST prevents PEP from determining if the remote is powered on, PEP can have no way of knowing whether this is a real transmission or just a glitch.

The remainder of this section applies if you're running PEP without using /NOTEST.

You may disconnect after a beep, before you start a new transmission.

At all times, PEP watches what's going on with the communications lines, and will know when something's amiss.

If there is a problem during a transmission (including the five-second pause after it), then PEP will consider this an error, produced by a loose cable or a hardware malfunction. This error condition may also be produced by turning off the remote computer during a transmission. So DON'T TURN THE REMOTE OFF DURING A TRANSMISSION.

Between transmissions, PEP will allow any number of disconnections and reconnections without objection. Any spurious data generated by a disconnect or a reconnect will be ignored by PEP, since PEP ignores data which is presented to your PC immediately before or immediately after either disturbance.

So, you can disconnect your cabling, turn on or off any of your remote hardware, or even change the remote computer without any problem (and without having to take remedial action at the PC keyboard), as long as you do this between transmissions. That means AFTER A BEEP, BEFORE STARTING ANOTHER TRANSMISSION. And PEP will still be running at the PC, ready to process the next file you send it.

Also, between transmissions and after the beep is the best time to restart PEP by pressing Ctrl-R, or to terminate PEP by pressing Ctrl-T. If you wait for the beep, you won't lose the remainder of an incoming file.

4.7 Transmission of multiple files

You'd like to transmit 51 diskette files from your remote computer to your PC, but you don't want to have to remain stationed at your remote's keyboard (or even in the same room as the remote). You want the 51 files to be distinct when they're saved on the PC. Can you do it?

Certainly, and it's very easy. Well, it's very easy if the remote has a "decent" operating system or you can program a bit.

An example : MS-DOS

As a hypothetical example, let's suppose that you've linked two PC's. We'll show you how to do an automatic multiple file transfer using MS-DOS batch processing and a very simple auxiliary program.

The auxiliary program should be one you've written which will essentially do nothing but waste time. It should waste more than five seconds, say fifteen seconds. (The actual program might be a simple loop written in BASIC, and might be loaded and executed using a DOS command like BASICA WASTE15.) We'll denote the DOS command which loads and executes this program, W15. (So W15 is just an abbreviation of BASICA WASTE15, or some other genuine DOS command.)

In whatever way seems easiest, create a diskette file which lists the files you want to transfer, one to a line. (For example, you might want to redirect the output of a directory listing to a diskette file, using a DOS command like

```
DIR *.BAS > PEPLIST.BAT
```

and then edit the file PEPLIST.BAT.) The file PEPLIST.BAT should then look something like

```
MAILLST.BAS
CUSTOM.BAS
NOTES.BAS
LABELS.BAS
INVOICES.BAS
```

Starting with this file, use a GOOD text editor to modify it until it becomes

```
COPY MAILLST.BAS AUX1:
COPY CUSTOM.BAS AUX1:
COPY NOTES.BAS AUX1:
COPY LABELS.BAS AUX1:
COPY INVOICES.BAS AUX1:
```


(Here we're assuming that the transmitting PC will use asynchronous card number 1.) Then edit again to obtain

```
W15
COPY MAILLST.BAS AUX1:
W15
COPY CUSTOM.BAS AUX1:
W15
COPY NOTES.BAS AUX1:
W15
COPY LABELS.BAS AUX1:
W15
COPY INVOICES.BAS AUX1:
W15
```

Save the final version of the file as PEPLIST.BAT. The actual name of the file is not terribly important, as long as it has the suffix .BAT so that the DOS will know it's a batch command file.

Now, executing this batch will result in the automatic file transfer you want. Just use the DOS command

```
PEPLIST
```

Incidentally, the initial and terminal W15 commands are not necessary, and may be removed from PEPLIST.BAT. Of course, you should have PEP running on the PC which is receiving the data, and the two computers should be in agreement on baud rate, etc. At the remote, use the DOS's MODE command to set the transmission protocols.

Another example: a small BASIC program

The same idea can be easily realized using almost any remote computer. If the remote can be programmed in BASIC, then you should be able to write a short BASIC program which

(a) Opens an RS232 port, and then executes a "waste time" loop to waste at least five seconds

(b) Obtains the name of a diskette file (from a DATA statement for example)

(c) Opens this file

(d) Reads a record from the file, and writes it to the RS232 port (first, check for end-of-file)

(e) Continues (d) until the end of the diskette file is reached, at which point the diskette file is closed

(f) Executes a "waste time" loop which is designed to waste fifteen seconds

(g) Repeats steps (b) through (f) until there are no more file names (or the file name is a dummy name like ZZZZZZ)

(h) Closes the RS232 port

(i) Terminates.

Even if your remote has a flexible operating system, this type of program can be very useful. It doesn't require all of the editing that we mentioned for the MS-DOS example. You could obtain the sequence of file names from a diskette file, somewhat improving the program.

4.8 Fine points

If you're using PEP to move a lot of data in one sitting, and you're transferring the files to a floppy diskette on the PC, then from time to time you might encounter a fatal error. This will happen when your floppy is full, and cannot continue to store the rest of an incoming file.

If your PC has a hard disk, the obvious remedy to this inconvenience is to transfer the files to it.

Otherwise, it is advisable to transfer the files to floppies which initially are "blank," that is, have no data on them. Try to keep track of the amount of data which you're sending, using a directory listing at the remote computer, and stop when you think that the PC's diskette is about 70 percent full. Between transmissions, swap diskettes at the PC and continue.

If you're doing a multiple file transfer onto a PC floppy, decide in advance how many files to transfer based on their collective size. Transfer a limited number of files, change the PC diskette, and then commence another multiple file transfer.

The BASIC program outlined in the preceding section requires some clarification.

First, let's suppose that you're transferring ASCII files (that is, readable text). For step (d) it is advisable to use a LINE INPUT statement to read the remote's data from diskette, rather than an INPUT statement. (INPUT statements get all confused when they encounter commas.) The string read from the diskette will NOT have a carriage return - linefeed terminator. When you transmit the string through the RS232 port in step (d), then for some BASICs it is the mode in which you have opened the port in step (a) which determines the characters appended to the string as it is transmitted. For text files, you want the appended terminator to be a CR-LF sequence, which for most computers will be the "default" situation.

On the other hand, PEP is quite capable of receiving binary files. Read such a file from the remote's diskette using an INPUT$ statement if the remote's BASIC recognizes one. (If not, you'll have to make do with a LINE INPUT or LINPUT statement.) When you transmit though the RS232 port, you DON'T want anything appended to the string as it is sent. For some BASICs you should open the RS232 port in a special mode which refrains from appending carriage returns or linefeeds. For other BASICs, you should have a trailing semicolon in the PRINT statement you're using in (d), and you should set the RS232 port to "infinite width" as the first step in (a).

PEP should receive binary files straight through (data handling option S) and it's a good idea not to display the incoming data at the PC.

In short, you'll need one program running at the remote to transfer text files, and a rather different one to transfer binary files. As well, you might need to learn about the modes in which the remote's RS232 port functions in BASIC.

NOTE 1: If you're doing a multiple file transfer for the purpose of transferring data to your PC's diskettes, DON'T PRINT THE INCOMING DATA.

Why? Suppose that you do try to print. Then because your printer is so slow, PEP's communications buffer will be filled rather quickly by the incoming data, and emptied rather slowly as the printout proceeds. Remember that to distinguish data packets, PEP will require a five-second pause in communications AFTER the buffer has been emptied. This could imply that the transmission must pause for several minutes between the transmission of separate files, in order to place these in separate diskette files on the PC.

NOTE 2: It would, however, be perfectly normal to print, and not save, multiple incoming files. This is because the pause is only meaningful when you're saving the files to diskette.

NOTE 3: If the transmitting computer is very slow, it would probably be preferable to use PEP's long pause option, L. You should then guarantee a pause of at least 30 seconds between files at the transmitting computer.

NOTE 4: PEP is very, very quick at receiving, displaying, and saving the incoming data. If you're not printing the incoming data, PEP should be able to empty the data buffer just about as fast as the incoming data will fill it.

We haven't ever seen PEP get ten seconds behind while it's just saving the incoming data, using as its data handling option list only S. In this situation, if you're using a PC with DMA capabilities (like an IBM PC) then getting ten seconds behind is fairly unlikely.

So, in our experience, a fifteen-second delay at the transmitting computer will be sufficient as an inter-packet separator (assuming that PEP is looking for five-second pauses, i.e. you're not using the L data handling option).

If you experience otherwise, here are some easy fixes:

If your transmitting software-hardware combination is so blindingly fast that PEP is getting behind and occasionally saving two separately transmitted files to a single PC file, first try to speed up PEP itself. Omitting the D data handling option should give you additonal speed, since displaying the incoming data is more time-consuming than just counting the characters. Use the conversion option S if possible, since this will eliminate the conversion of the incoming data before it is stored on diskette.

Try speeding up your diskette hardware. Use a hard disk or a virtual disk, rather than a floppy, to store the incoming data.

Finally, if necessary, slow down the transmission. You might try a slower transmission speed, say 4800 baud. That should do it. We don't think that this will be required, but we may not have pushed PEP to its limit in our testing.

Alternatively, you can slow the transmission by using a smaller buffer, say 1K. This will cut the transfer speed somewhat, but certainly not by as much as using 4800 baud.

(If you're doing multitasking, like running a background print spooler, this slows PEP. Stop the background processing, and try again.)

NOTE 5: If your remote computer's BASIC occasionally halts while it performs a "garbage collection" of dynamic string variables, then the maximum period of time which elapses during the garbage collection should be taken into account when you determine the inter-packet pause interval.

If the garbage collection is time-consuming, use the long pause option L to be on the safe side. Of course, this applies only to situations in which you're using a BASIC routine to output your data.

Notes

Chapter 5.

Error Conditions

PEP accepts whatever is satisfactory to the hardware in the PC's asynchronous adapter, and processes this data very simply. PEP itself makes no checks on incoming data, and therefore produces almost no error messages which could properly be called its own.

However, PEP is a prisoner of your hardware and operating system ("DOS") and must faithfully announce error conditions as reported by the system.

5.1 The Critical Error Handler

We assume that you're reasonably familiar with the DOS's "Critical Error Handler," which is DOS's way of indicating a persistent hardware problem. It will produce messages like

```
Not ready error reading drive A
Abort,Retry,Ignore?
```

This error message is produced directly by the operating system without PEP's knowledge. It allows you to take remedial action, such a placing a diskette in drive A, to allow another attempt at input/output with this drive. You should of course press R if you wish to perform such a retry.

Consult Appendix A of your DOS manual for more details.

5.2 Errors during program load

PEP is divided into two separate modules, which are called PEP.COM and PEP.OV on your program diskette.

PEP.COM is the "shell module." It is this program which is immediately loaded and executed when you enter the DOS command PEP. If you see the message

Bad command or file name

when you're attempting to load PEP, then the file PEP.COM cannot be found by the command processor.

PEP.COM is responsible for establishing a special environment in which RS232 communications are given a very high priority ("interrupt-driven"). PEP.COM also assigns space (a "buffer") in the computer to receive and temporarily store the incoming data. PEP.COM will also locate PEP.OV, load it into computer memory, and begin executing it. PEP.COM will require that you're running DOS 2.0 or higher before it attempts to do any of this.

Incidentally, when you terminate PEP (even via the critical error handler) then control is passed to PEP.COM, which restores the computer's original environment.

PEP.COM may encounter several difficulties in getting things rolling. If you're not running the correct version of DOS, you'll see the error message

Incorrect DOS version
— PEP requires 2.0 or above !

PEP.COM may not be able to find room for the buffer, in which case the error message

ERROR : Error with memory allocation

will appear. If PEP.OV cannot be found (looking first in the current directory and then in the root directory of the default drive) then you'll see

ERROR : Unable to locate PEP.OV

And if the DOS cannot load and execute PEP.OV, you'll see

ERROR : Unable to execute
child process PEP.OV

Each of these error messages will terminate PEP, and return you to the command level of the DOS.

5.3 Reaction to load errors

You may take corrective action in response to each of these errors.

If you see the "Bad command" error message as you begin to load PEP, you should place a copy of PEP.COM in the current directory of the disk you're using, change the current directory to be one which contains a copy of PEP.COM, or move a copy of PEP.COM into one of the directories which the command processor searches. (See the PATH command documentation in your DOS manual for further details.)

If the "Incorrect DOS" message appears, you need a new DOS, Version 2.0 or higher.

If there is a memory allocation message, then you're out of memory in the computer. You might try loading PEP with a smaller buffer, using a command like

PEP /B = 1

However, all by itself this measure probably will not be of much help. DOS will still prevent you from loading PEP.OV since only 31K of computer memory will be freed by shrinking the buffer to 1K, and this isn't enough room for PEP.OV.

If you're using memory-hungry resident utilities (like a RAM disk or special device drivers or extensive desktop utilities) then you should try loading PEP without these in the system. (You'll probably need to change the configuration file, CONFIG.SYS, to do this. Then restart the system with Ctrl-Alt-Del.) Or, you might need to install more memory in your PC. To the best of our knowledge, PEP can be made to work with any PC having 192K or more of installed memory.

If PEP.COM can't find PEP.OV, this is because you don't have a copy of PEP.OV in the current directory, nor is there a copy in the root directory. Put a copy in the root directory (of the default disk).

Lastly, if PEP can't execute PEP.OV then this is probably because you're out of memory. Follow the advice given above, in reference to the memory allocation error message. Start by trying the /B = 1 command line option as you load PEP.

5.4 Load options

PEP will not report any errors regarding your use of command line options when you load the program.

If PEP encounters an option of the form /B = nn, and nn is one of the numbers 1, 2, 4, 8, 16, or 32 then PEP will reset the size of the communications buffer being used. If nn is not one of these numbers, PEP will use a 32K buffer, and will not give you special notice regarding this interpretation. However, the buffer size is displayed as PEP runs, on the upper horizontal bar.

If PEP encounters an option /D, then the PC's Data Terminal Ready pin will be controlled in order to enable and disable data transmission from the remote computer. Otherwise, PEP will assert control over the Clear to Send pin.

The command line option /NOTEST will cause PEP to refrain from testing the PC's Data Set Ready pin, to determine whether the remote is connected and powered on.

PEP will ignore any other expressions in the command line.

5.5 Communications errors

Most of the errors you'll encounter during PEP's execution will concern the RS232 communications between your computers.

These error messages will appear between the two horizontal bars on the screen. Each will be preceded by two solid rectangles.

Between transmissions, when the remote computer's communications hardware is unavailable (for example, because the remote is turned off), PEP will display the warning message

COMMUNICATIONS CHANNEL UNAVAILABLE

and will idle until the hardware is available. PEP tests the asynchronous adapter you've specified for signs of life, and so this message will appear if you've specified the wrong asynchronous adapter at your PC. Since this is merely a warning message, you need not take any action at the PC in response to it. (If you're using the /NOTEST command line option then PEP will not produce this message, and will not know when the remote is available.)

While you're receiving data, if the transmitting hardware becomes unavailable then PEP will display

ERROR : LINE FLUCTUATION

Again, if you're using /NOTEST then this message won't appear.

If the incoming data cannot be interpreted by the PC's hardware (wrong baud rate, stop bits, etc.) then you'll see a message of the form

DATA ERROR: Break interrupt Framing error

except that perhaps only one of the error categories ("Break interrupt" or "Framing error") will be mentioned.

When one or both of these error messages appear, you will decide how to proceed. You'll see the prompt

Press Ctrl-T to terminate,
Ctrl-R to restart program,
Ctrl-G to continue data reception, or
Ctrl-I to ignore further data errors,
this file (and continue)

just below the error message, and you should press one of the four indicated keystrokes to proceed.

5.6 Interpreting communications errors

When you encounter an error message of the type listed above, it is usually appropriate to determine the source of the error before proceeding.

Line fluctuations are caused by loose connections, loose wires or connectors or "shorts" inside the cabling, or intermittently faulty hardware. If the connections have loosened, tighten them carefully and then retransmit the file which experienced the error. If you suspect faulty hardware, then try using both computers to run different communications applications, for example, communicating through a modem to a mainframe.

Although it is rather unlikely, the software running on the remote computer may also be causing line fluctuations. Normally, the sending hardware should keep its DTR pin on while the remote hardware is operational, and this condition is tested at PEP's DSR pin. If the sending software is turning DTR on and off, then there will appear to be a problem. (Is your cable ok? Make sure the remote's DTR pin is wired to the PC's DSR pin.) It is for precisely this situation that the /NOTEST option is available, and you should use /NOTEST with problematic software.

The message ERROR : LINE FLUCTUATION will also appear if you turn off the remote computer while PEP was receiving and processing incoming data. Remember that you should wait until the PC beeps before turning off the remote or transmitting another file.

The DATA ERROR messages result when an incoming data stream is not constructed in the way the PC expects. This is usually the result of specifying different baud rates or a different number of stop bits to the two computers. Incidentally, "Break interrupt" usually occurs when the remote is sending at a baud rate lower than the PC expects. "Break interrupt" is an indication that the PC has just received an impossibly long sequence of "1 bits."

If you have a DATA ERROR, you should make certain that the two computers are attempting to communicate using the same protocols. This includes the number of data bits, the number of stop bits, and the baud rate. As well, some hardware will sometimes attempt to send an additional bit if there are 8 data bits; a clause like "PA = N" should turn off this feature at the remote. Incidentally, PEP does not make a parity check of 7 bit data; it will correctly receive even or odd parity data without any noticeable preference or change in performance, and without any check for correct parity.

You should be able to modify the communications protocols being used at the remote computer by the sending software. If not, you should probably use other software, rather than attempt to discover what the approriate protocol is. (This may ultimately be a trial and error process at the PC.)

5.7 Proceeding after an error

If you're just beginning to use PEP, then it is acceptable to press Ctrl-G (that stands for GO) or Ctrl-I when you encounter errors.

Ctrl-G will cause the single condition which has caused the error to be ignored.

Ctrl-I will cause PEP to ignore the current error. As well, during the transmission of the present data packet, PEP will ignore all further conditions which would cause a DATA ERROR. (If you're using /NOTEST then this results in ignoring all transmission errors.)

However, once you're past the experimentation stage, you should usually choose one of the other courses of action. The incoming data should be considered unreliable if any error has been detected in its transmission, and so the data file should be retransmitted. Both Ctrl-R and Ctrl-T are appropriate, but you will probably press Ctrl-T if you're experiencing a line fluctuation or if the cause of the error is not immediately obvious. Remember, however, that if you terminate PEP then you should rename or move the files REMOTE.XXX before loading PEP again.

Remember also that an invalid data file may have been written to your diskette, and you should note its name before proceeding if this is the case. Then you'll be able to erase it later, without confusion.

5.8 Other errors

Whenever PEP attempts to write on a diskette or print to your printer, it checks to see if the DOS has reported any problems.

For example, it may be that the diskette is full or that it is write protected, or that the diskette file REMOTE.XXX is not available to be written upon since it is marked "read only."

If the error has occurred when attempting to write to diskette, you'll see a message of the form

Disk file I/O error :
code numbers iii nnn for file REMOTE.XXX
Press any key to terminate the program

This message indicates the name of the file, REMOTE.XXX, which can't be written to, and the numbers iii and nnn provide information about the nature of the error and the progress made in writing to this file. The interpretations of the values for iii are:

1	File does not exist
3	File not open for output
145	Seek beyond end-of-file
153	Unexpected end-of-file
240	Disk write error
241	Directory is full
242	File size overflow
255	File disappeared

Incidentally, PEP.OV is written in Turbo Pascal, and receives the error numbers iii from the Turbo Pascal I/O procedures. The number nnn is interpreted as

1	Assign
2	Open
3	Write
5	Close

If PEP is having problems with the printer, a message of the form

Printer error : code number iii
Press any key to terminate the program

will appear. The number iii will indicate the error type, interpreted as above, and again provided to PEP by the Turbo Pascal I/O procedures. Remember that PEP uses the system device PRN: (also known as LPT1:) as the printer.

As you've surmised, the errors described in this section are fatal. You must respond to an error by pressing a key, which will terminate PEP.

If you should find yourself in the unlikely situation of sending thousands of files to be saved on your PC, PEP can't handle this in one run. Remember that PEP will automatically terminate when the file REMOTE.999 has been closed. Not an error, but certainly an irregularity!

5.9 A special error message

An inexperienced user may attempt to execute the overlay file PEP.OV directly, without having it loaded by PEP.COM. PEP incorporates facilities to prevent this, and will produce the message

Error : PEP.OV cannot be executed

if PEP.OV is executed directly. (This is only possible if PEP.OV has been renamed, since the current DOS ordinarily won't attempt to execute files with a file extension .OV) You'll be returned to the DOS, without any changes in the system environment.

Notes

Appendix A.

Interfacing a Remote Computer to Your PC : What PEP Requires

Why all the technicalities?

Appendices A, B, and C will help you obtain a cable to connect your computers. We'll tell you right off the top that this is probably a "standard null modem cable," available at most Radio Shack stores.

Except . . . if you're connecting your PC to a TI-99/4A or to a standard modem, the appropriate cable is probably "straight through."

In these appendices, you'll find some rather complicated technical information. This information is about the nature of the communications standard (or "protocol") called RS232. RS232 is the most widely available communications standard for microcomputers; the latest version of this standard is called RS232C.

We'll tell you a bit about RS232, the hardware on the PC, the communications hardware on your remote computer, and the cable you'll need to connect the two computers.

RS232 communications are rather difficult to understand, and even more difficult to explain. As we mentioned above, you'll probably find that a standard null modem cable will suffice to connect your computers, and you won't have to understand a single word of what follows.

If you didn't understand our explanation of PEP's command line option /D in Chapter 1, that probably won't be a problem either. Most likely, PEP will run perfectly for you without this option, especially if it turns out that you'll be using a standard null modem cable.

So, why all the technicalities? Unfortunately, a number of problems may arise in attempting to connect your computers. We provide the details in the appendices to help you overcome these problems, and to help you to speak knowledgeably to your technician.

The asynchronous adapter

The standard RS232 card which is installed in most PC's is called an "asynchronous communications adapter."

Attached to the rear of this card, and projecting through the rear of your PC, is a connector. The connector is a standard RS232 25-pin male D-shell connector. It's clamped to the housing of your PC.

To communicate with devices like your remote computer, you'll need to attach an appropriate cable to the connector on the asynchronous adapter. On the PC end, the cable should terminate with a standard RS232 25-pin female connector.

Gender

There are two general kinds of RS232 connectors. One of them has 25 or so visible pins, and is called a male connector. The other kind has 25 small holes, and is called a female connector. They are designed so that connectors of different gender can be plugged together.

Female connectors are also called "socket connectors," and sometimes referred to by the abbreviation DB25S. Male connectors are also called "plug connectors," or DB25P.

You should carefully note the gender of the connectors on your computers, so that you can obtain a cable whose connectors are of the correct gender.

In what follows, our discussion of "pins" is not an attempt to refer only to male RS232 connectors. Rather, the term "pins" is intended to refer to the "positions" on an RS232 connector, be it male or female. These positions are labeled, by number, on most RS232 connectors.

The adapter's pins

To help you understand the details of obtaining the correct cable, it will be useful to know the function of the pins on the asynchronous adapter. The function of these pins can be summarized as follows:

Pin		Function	Direction
1	*	Protective ground (PG)	—
2		Transmitted data (TD)	From PC
3	*	Received data (RD)	To PC
4	*	Request to send (RTS)	From PC
5		Clear to send (CTS)	To PC
6	(*)	Data set ready (DSR)	To PC
7	*	Signal ground (SG)	—
8		Carrier detect (CD)	To PC
20	*	Data terminal ready (DTR)	From PC
22		Ring indicator (RI)	To PC

Excuse the vocabulary. Most of these terms are at least mentioned in the PC's technical reference manual. If you're interested in what this all means, you should read "RS232 Made Easy" by Martin Seyer (Prentice-Hall).

Incidentally, "carrier detect" is sometimes called "data carrier detect" or "receive line signal detect."

I find the terms "from" and "to" rather confusing, and prefer to think of which pins are controlled by the PC and which are not. "From PC" then means controlled by the PC. "To PC" means controlled by the peripheral, transmitting data to the PC or being tested by the PC.

Some of the other pins on the asynchronous card are meaningful, but not appropriate to our discussion.

The pins are classed into three general groups. Pins 1 and 7 are "grounds," and should always be attached to the corresponding pins (almost always numbered 1 and 7) of the remote device in order to protect your equipment. Pins 2 and 3 are "data" pins, used to transmit data to and from the asynchronous adapter. The other pins mentioned in the above diagram are "control" pins, used to control the direction and timing of data transfer.

Thinking about pins

You just want the correct cable. You're probably thinking about lines (or wires), not pins.

You may also find it strange to see a pin which is physically attached to the PC being described as "controlled by the remote computer."

However, it really is better to talk about pins than about lines. The pin on your remote computer which transmits data will be attached to a line which is connected to your one of your PC's pins. This pin will be the one which receives data.

What would we call such a line? The "remote to PC data transmission line"? Not such a good concept, since the notion of "remote" is relative, and RS232 terminology was around long before there was a PC.

It would be better to call one of the pins at the remote a "data transmitting pin" and one of the pins at the PC a "data receiving pin." That's what we've done, just using different words.

And of course, the PC's data receiving pin really is controlled by the remote computer, even though it's attached to the PC. The remote computer controls its data transmitting pin electrically, and this control is applied to the PC's data receiving pin by the wire which conducts the electrical current, thereby controlling the PC's data receiving pin.

Got the general idea?

The pins which PEP requires

In the above table, the pins which PEP uses (or are required to complete an adequate RS232 circuit) are marked with an asterisk.

Pin 6, the Data Set Ready pin, is marked (*) to indicate that it is optional. If you will always be running PEP using the /NOTEST option then PEP will not test this pin, and its connection will not be required. (Of course, it is not recommended to run PEP using /NOTEST, unless you believe that you have no choice. So, you should almost certainly connect pin 6.)

A lot of PC software ignores Ring Indicator (pin 22) and Carrier Detect (pin 8), and this is the case with PEP. Since PEP doesn't actually transmit any data of its own, it doesn't use Transmitted Data (pin 2). PEP ignores Clear to Send (pin 5).

Pins controlled by PEP

Two of the PC's RS232 control pins, Request to Send (RTS) and Data Terminal Ready (DTR), are actively controlled by PEP. Ordinarily, both of these pins should be wired to the remote computer, and the status of at least one of them should be tested by the remote before it attempts to transmit data to the PC.

PEP keeps both of these pins "on" most of the time. One of them will always be kept on, and the other will occasionally be turned off to prevent the remote computer from transmitting data.

If you use PEP without the /D option, then PEP will always keep DTR on, and will use RTS to control the remote computer. PEP will only turn RTS off to prevent the remote from transmitting. In this situation, the RTS pin must be attached to the remote computer, and the remote must test its status before transmitting data. Usually the RTS pin is wired to the remote computer's Carrier Detect pin.

If you use the /D option, then PEP will always keep RTS on, and will use DTR to control the remote computer. PEP will turn DTR off to prevent transmission from the remote. In this case, DTR must be wired to and be tested by the remote. Usually the DTR pin is wired to the remote computer's Data Set Ready pin.

The /D option is available to enable PEP to function with some unusual hardware, like the TI-99/4A Home Computer, using fairly standard cabling. It's a rather bizarre option to handle a rather bizarre computer.

PEP will turn RTS (or DTR if you're using /D) off during Phase II of its operation, under four different circumstances:

(1) when you press Ctrl-H to pause the transmission. When you press another key to resume transmission, the pin will be turned back on.

(2) when PEP detects an error. You will be required to indicate a subsequent course of action by pressing a key at the PC. Once you make your selection, PEP will put the pin back on.

(3) when the input buffer is nearly full. The pin will remain off until there is room for at least 300 additional characters in the buffer, at which point it will be turned on again.

(4) Also, off will be maintained for a period of time after PEP detects a disconnect between data transmissions. The off condition will persist until there is a reconnect, and a test of the pins controlled by the remote computer indicates that they're continuously normal and stable for five seconds after the reconnect. At this point the pin is turned back on.

Pins tested by PEP

The PC's Data Set Ready (DSR) pin is continously tested during Phase II of PEP's operation. (Exception: If you're using /NOTEST, then DSR is ignored. We'll assume that you're not running PEP with the /NOTEST option in what follows.) None of the PC's other control pins are tested by PEP.

Provided that you're using the right kind of cable, DSR normally tests ON when your remote's communications hardware is functioning. In most cases, this means that DSR will be turned on by the remote computer when it is powered on, and it will remain on until you turn off the remote.

(Here, the "right" cable connects the remote's Data Terminal Ready pin to the PC's DSR pin. So, your remote computer should turn DTR on, and keep it on, until you turn off the remote. If this isn't the case, either connect the PC's DSR pin to a pin on the remote's RS232 hardware which the remote's software leaves on when the hardware is operational, or as a last resort run PEP with the /NOTEST option.)

(Again, turning "on" is opening a communications channel, and turning "off" is closing the channel.)

Between transmissions of data packets, if DSR goes off then PEP interprets this as a normal and deliberate disconnect. PEP will indicate on the screen that the communications channel is unavailable, but you need not take any remedial action at the PC in response to this. If DSR is later detected ON, then PEP will wait for five seconds to stabilize the line, then indicate on the screen that the communications channel is available, then beep, and then start accepting any incoming data as a valid transmission.

During the transmission of a data packet (including the five-second pause after the transmission is complete), DSR is expected to remain on. If it goes off, even for a tiny fraction of a second, then this is interpreted by PEP as a transmission error. (Incidentally, this is most likely caused by a loose cable or an intermittent hardware fault in one of the computers. Or, by a user who has turned off the remote computer at an inappropriate time.)

Lastly, PEP will use Received Data (RD) to receive the data transmitted by the remote computer. This pin should be wired to the remote's Transmitted Data (TD) pin.

Usually, any data entering the PC through this pin while DSR tests ON will be interpreted as a valid transmission from the remote computer. (The only exception is while PEP is restabilizing after a disconnect — incoming data will be considered erroneous during this period.)

NOTES

Appendix B.

Interfacing a Remote Computer to Your PC : Making a Cable

IMPORTANT NOTE

It is difficult to be absolutely certain that a particular cable will function perfectly with all of the software on your remote computer.

If, for example, the remote software turns off the remote's Data Terminal Ready pin from time to time, you have a minor problem. PEP will think that there is a disconnect, since at the PC the pin Data Set Ready will go off.

Fortunately, most software will leave DTR on, since this is the "master switch" that is usually used to indicate that the communications hardware is operational. Any software which turns DTR off would, in effect, be disconnecting ("hanging up").

You should run PEP with the /NOTEST option if you're getting unexpected disconnects.

Incidentally, even PEP ignores the convention about maintaining DTR on, when you run PEP using the option /D.

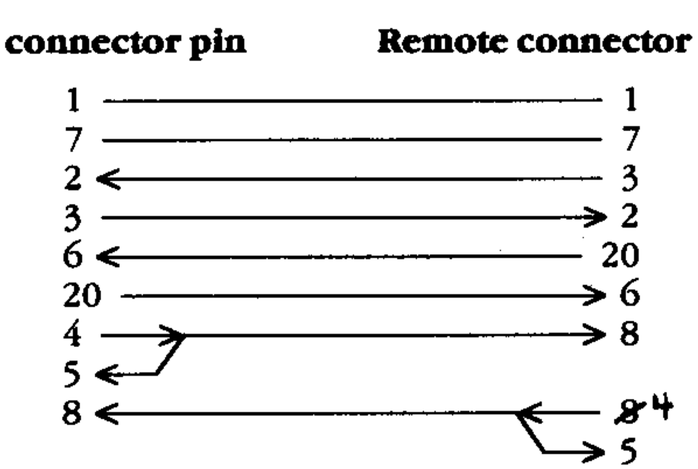
Connecting your PC to the remote: an example

Most computers use the RS232 pins exactly as your PC does, for example the Commodore VIC-20.

An excellent cable to connect your PC to a remote whose pins have the same meaning as the PC's, is a "standard null modem" cable. It's a bit tricky, especially if you're doing the work yourself, but this is probably the best way to go. It should work with PEP, and probably any communications software you ever run between the two computers. And you should be able to get one inexpensively, right off the shelf.

The "standard" cable pays close attention to the exact meanings which the RS232 protocol assigns to the RS232 pins, and should work with virtually any software running on your two computers. We won't go into this — if you're interested, consult Seyer's book.

Graphically, a standard null modem cable is wired as follows:



The arrowheads indicated in the above table are used only as a reminder as to which computer is transmitting through, or controlling, the pins.

Your cable should connect 1 to 1, 7 to 7, and cross connect the pins 2 and 3 and the pins 6 and 20.

However, the connection of pins 4 and 5 is quite different — they should be connected together right at each of the RS232 connectors rather than between the two computers. (REMEMBER, however, that we're still talking about the "ideal example" in which your PC is connected to a remote computer whose RS232 pins mean EXACTLY THE SAME THING as the PC pins. Fortunately, most computers have at least one communications port which satisfies this criterion.)

Additionally, pin 8 gets into the act in case you use an application which tests it. Pin 8 is wired to pin 4 of the OTHER computer. (Or to pin 5, since pins 4 and 5 are wired together.)

Note that the cable itself is perfectly symmetrical — the labels identifying the connecting computer are not really significant.

If you're lucky enough to have a VIC-20, then you can now go out and purchase your cable, since your technician will probably want to give you a standard null modem cable or will want to know the exact pin numbers which are to be connected to make one. (Remember that you'll have to know the gender of the RS232 connectors on your communications hardware for each computer.)

A number of companies, including Radio Shack, manufacture "null modem adapters." These are compact RS232 plugs which when attached to a straight through cable convert the cable to a standard null modem cable. A null modem adapter is an economical alternative to purchasing a null modem cable, if you already own a straight through cable.

In fact, according to Seyer's book, the cables we have described will apparently work with any of the following computers :

Adds Multivision
Apple III
Apple II, with Super Serial Card, Port : Modem Position
Atari 400/800, Port 1 on 850 Interface
Burroughs B21 or B22
Commodore Super PET (9000) and VIC 20
Convergent Technologies 1000
Corvus Concept
Cromemco Quadart, Ports J3, J5, J7, and J9
Data General MPT/100
Datamac 800/1600 — Port A
Digital Equipment Corp. Rainbow PC-100,
  Port : Communication
Epson MX-20
Heath H-8, Port : DCE
Heath H-89A or Zenith Z-89, Port : DTE
Hewlett-Packard HP125, Port : 2
Hewlett-Packard Series 80

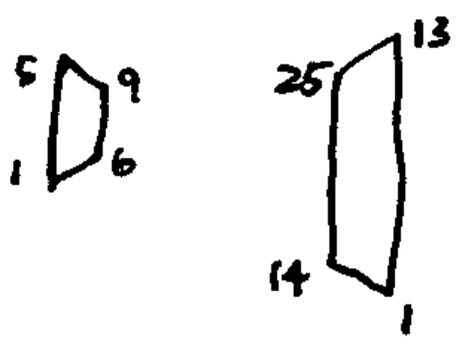
PEP Cable 27 Dec 86

Description of Cable A page C-3 for connecting a regular PC to TI-99/4A without using PEP/D.

AT PC				99/4A
X	#1	Protective Gnd.	—	Protective Gnd.
5	7	Signal Gnd	—	Signal Gnd.
3	2	Transmitted Data	→	2 Received Data
2	3	Received Data	←	Transmitted Data
8	5	Clear to Send	←	5 Request to Send
6	6	Data Set Ready	←	Data Terminal Ready
7	4	Request to Send	→	Data Set Ready
1	8	Carrier Deted	←	8 Data Carrier Detect
4		DTR	→	

See p. A-3 for these descriptions of the Standard PC. See p. 8 of READ.ME file for description of AT pins. See p. C-3 for description of TI pins (also see p. 26 of TI RS232 Interface Card manual).

With this cable you should use PEP. PEP will use RTS pin on PC or AT to signal TI when to send data. See p 1-5, 1-6 of PEP manual.



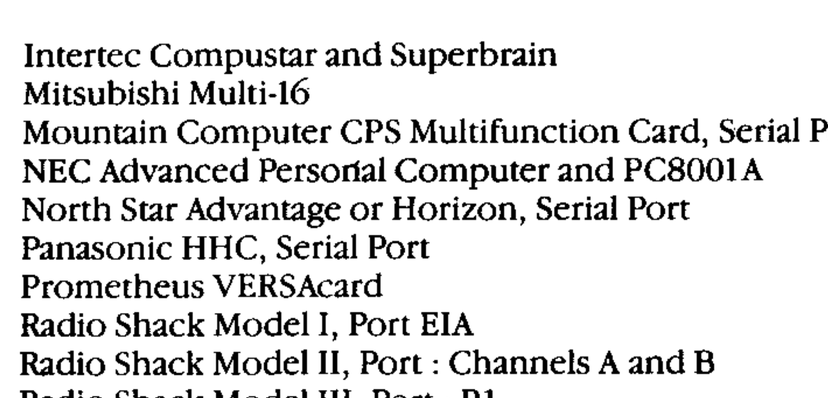
TRANSMIT FROM TI/99-4A VIA
RS232.BA=9600.LF

Intertec Compustar and Superbrain
Mitsubishi Multi-16
Mountain Computer CPS Multifunction Card, Serial Port
NEC Advanced Personal Computer and PC8001A
North Star Advantage or Horizon, Serial Port
Panasonic HHC, Serial Port
Prometheus VERSAcard
Radio Shack Model I, Port EIA
Radio Shack Model II, Port : Channels A and B
Radio Shack Model III, Port : P1
Tarbell Empire, Port A/B on I/O Board
Televideo TS800A, TS802, TS80H, Port : P1 (DTE)
Texas Instruments Business System 200
Toshiba T100/200/250, Port : Serial
Xerox 820/820-2

A general example

As you may already have surmised, all this attention to pin numbers is obscuring the principle of the thing.

If you don't have one of the computers listed above, then for the recommended null modem cable the wiring should conform to the following principle:



Does that solve everyone's problem?

Not quite. The above suggestions should work well if all of the signals we've outlined on the PC's serial card happen to be present on the remote serial card as well, and you can determine which pins on the remote have which function.

However, some computers (like the TI-99/4A) aren't so lucky. As well, you may want to apply PEP to data entering your PC via a modem. More on this in Appendix C.

Appendix C.

Interfacing Your PC to a Modem or to a TI-99/4A

The cable

To make a long story short, a straight through RS232 cable will probably do the trick. ("Straight through" means pin X is attached to pin X.) For this cable to be generally useful, lines 1, 2, 3, 4, 5, 6, 7, 8, 12, 20, and 22 should be present.

Straight through cables are available right off the shelf at most Radio Shack stores. If both ends of the cable are of the same gender (most likely both male), you need a straight through adapter to reverse the gender of one end of the cable; again, you can get this at Radio Shack.

If you already have a cable which is used to connect your PC to a modem, this is almost certainly "straight through," and if so it will probably work perfectly.

Connection to the 99/4A

If you're interfacing to a TI-99/4A Home Computer, then the straight through cable should be male on one end and female on the other end. PEP will work just fine if only lines 1, 3, 6, 7, and 20 are present. You may omit line 6 if you intend to always run PEP with the /NOTEST option.

IMPORTANT NOTE : IF YOU ARE INTERFACING TO A TI-99/4A THEN YOU MUST RUN PEP USING THE /D OPTION WITH THIS STRAIGHT THROUGH CABLE.

You can use this cable to attach your PC to either connector of the 99/4A RS232 module, or directly to the connector on the RS232 expansion card in TI's expansion box. If you're connecting directly to the expansion box, then this links the PC to RS232 port number 1 of the 99/4A. If you have a short "Y-cable," and it's been built to the proper specifications, then using this with the straight through cable you can connect your PC to either 99/4A port (number 1 or 2).

So, what's the long story?

Glad you asked.

If you're still having trouble obtaining a cable for your computer, it might be useful to you to see just how difficult it can be to figure out what the correct cabling is. The TI-99/4A is a perfect example. Perhaps some of the following details will even be specifically helpful.

The 99/4A has two standard (Texas Instruments) RS232 attachments, one called a "module" and the other called a "card." With either one of these, TI tells us that "Port 1" has its pins configured as follows:

Pin	Function	Direction
1	Protective ground	—
2	Serial data into RS232	To TI
3	Serial data out of RS232	From TI
5	Clear to send for RS232	From TI
6	Data set ready for RS232	From TI
7	Signal ground	—
8	Data carrier detect for RS232	From TI
20	Data terminal ready for RS232	To TI

We're trying to quote TI verbatim here, hence the rather wordy description of "function."

Note that pins 2 and 3 appear to be reversed from the way that most micros use them, and that at first glance there seems to be no Request to Send pin. If you look even more closely (comparing this with the IBM pin descriptions), the direction of data transfer doesn't agree at all with the pin terminology. Hmmm.

The fact is, that TI has arranged the 99/4A pins as if the 99/4A is a modem ("DCE") rather than a typical computer ("DTE"). TI doesn't mention this, and it took quite some time to deduce. Once known, a straight through cable is the most likely candidate.

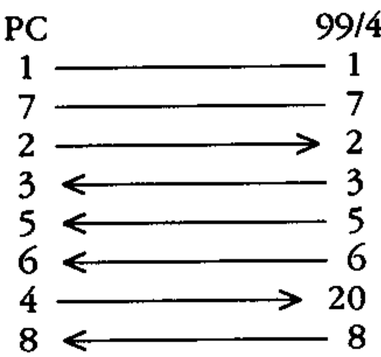
Our first attempt at unsnarling this description of the 99/ 4A's RS232 pins, was to suppose that the last column was correct and see what we came up with. We figured that (aside from its first two entries) the middle column might be properly described as the exact opposite of what the descriptions were. We wound up with something like this, using the standard RS232 vocabulary :

Pin	Function	Direction
1	Protective ground	—
2	Received data	To TI
3	Transmitted data	From TI
5	Request to send	From TI
6	Data terminal ready	From TI
7	Signal ground	—
8	????	From TI
20	Data set ready	To TI

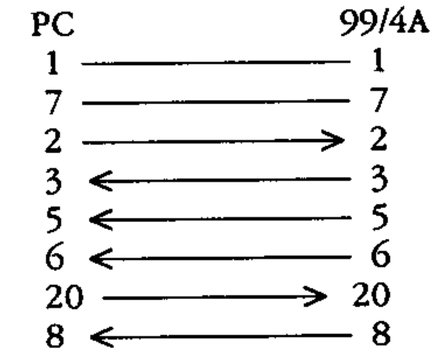
There appears to be no Clear to Send. Note that the TI has two data pins, one for outgoing data and one for incoming data, and that there are four control pins. Three of these pins are controlled by the TI, and only one of the control pins is tested by the 99/4A.

If this isn't quite right, at least this much insight enabled us to come up with two different cables which actually work:

CABLE A



CABLE B



That's right — cable A is almost straight through, and B is a straight through cable. Either one will work just fine, but B should be cheaper. You must run PEP without the /D option using cable A, but you must use the option /D with cable B. In fact, one motivation for providing the option /D is to allow you to use cable B with a 99/4A.

If you're making cable B, we recommend running lines 4, 12, and 22 straight through the cable. This isn't mentioned in the above diagram since the 99/4A doesn't use these pins.

A final note

This Appendix might annoy some RS232 purists. They would claim that it is nonsense to compare DCE (like the 99/4A) to DTE (like the PC). They have a point. We hope that some readers will find our explanation useful, nonetheless.

Although the TI hardware is rather strange, TI software tends to be very predictable. Aside from Terminal Emulator programs, most software will leave all of the control pins at the TI continuously on. In fact, TI doesn't even document how to turn any of these pins off. The only control pin which the TI can test is number 20, and the system software does test this pin before transmitting.

Appendix D.

Receiving COMPANION Printouts: The Conversion Option X

This appendix is intended exclusively for TI-99/4A users who are using Intelpro's word processor, COMPANION.

If you intend to receive and save a file which is being printed by COMPANION, you should select the conversion option X. As mentioned in Chapter 2, this will put linefeeds and carriage returns in the appropriate places for MS-DOS, and will replace every formfeed character with a backslash (\). For COMPANION printouts, this will have the effect of replacing the formfeed characters as well as most of the top page margins by a backslash character (\) in the first column of the line in which the formfeed appears.

The option X will make the underlining as printed by COMPANION visible. You may or may not appreciate the way that this is done. If you don't like the way that the conversion handles underlining, you can elect to suppress underlines at the 99/4A by removing the left underline brackets from your COMPANION text before printing it to the PC. (Use a global search and replace.)

REMEMBER : You should NOT elect to use the data handling option P ("print") in conjunction with the conversion option X, since the printer would not receive the correct formfeed character to make the printout acceptable. If you want to print an incoming COMPANION file, use the conversion option S.

NOTES

Appendix E.

Technical Notes

The program modules PEP.COM and PEP.OV communicate with one another using a mechanism called "software interrupts." They use interrupts numbers 96 and 97. These are saved and changed by PEP, and reset to their previous values when PEP terminates.

These interrupts are designated as "user interrupts" by the DOS, and consequently they should not be used by any resident utilities which you may have installed in your PC.

However, it is remotely possible that some utilities have illegally made use of these interrupts, and that they may therefore not function properly while PEP is running. Even worse, it is conceivable that a utility will be using and also changing these interrupts when it is called upon, causing PEP to crash.

PEP uses software loops, rather than a consultation with the hardware clock, to determine the duration of a pause between transmissions. If you're running PEP on a PC which operates at a clock rate in excess of 4.77 MHz (the clock rate of a standard IBM PC) or uses a faster processor than the Intel 8088, then PEP will appear to recognize shorter pauses as "inter-packet markers." You may elect to use the "long pause" option, L, to require a longer pause between transmissions, probably much less than 20 seconds for your machine.

If you're executing a background application on your PC, such as a print queue, then you may find that the required pauses between transmissions are somewhat longer than expected.

If you're using a PC with "extended memory," then you shouldn't run PEP while a virtual disk is residing in this portion of RAM. Evidently, interrupts are disabled when the extended memory is addressed, causing the loss of incoming data.

PEP has been developed on an IBM PC-XT, with 640K of RAM and an AST expansion card. The authoring machine has two asynchronous adapters and a monochrome monitor. PEP has been tested on the authoring machine and on an IBM-PC, with 256K of RAM, an AST card, one asynchronous adapter, and an Enhanced Color Monitor card.

PEP is not terribly hardware dependent, except in its interrupt-driven communications facilities. Any PC clone which is using the same communications chips as the IBM-PC, at the same port addresses, should be able to run PEP. Some hardware may require the installation of "PC emulation" software before running PEP; this emulation software should at least allow the PC to run Turbo Pascal in its IBM version.

Communications hardware details

Some specific hardware details may prove helpful. PEP expects that your PC is sufficiently similar to the IBM-PC that it uses essentially the same communications chips, the same I/O addresses, and the same IRQ request lines for the communications chips. These are:

Port Configuration	I/O Addresses	IRQ Line
COM1	3F8-3FF Hex	IRQ4
COM2	2F8-2FF Hex	IRQ3

Interrupts generated by COM1 are expected to be number 0C hex, vectored through addresses 30-33 hex, and COM2 should use interrupt 0B hex vectored through addresses 2C-2F hex.

The asynchronous adapter should use the Intel INS8250 LSI Asynchronous Communications Element (this is a communications chip) or its functional equivalent. PEP will enable interrupts on the communications chip by loading its internal interrupt register (at 3F9 hex or 2F9 hex) with an appropriate value (01 hex), and by setting the OUT2 bit in its Modem Control Register (at 3FC or 2FC). You may need to consult your hardware schematics to verify some of this; the IBM Technical Reference Manual for the PC may also be helpful.

With certain asynchronous communications adapters, it may be necessary to connect all of its inputs (excluding perhaps Ring Indicator, pin 22) to a signal, even if PEP appears not to use the input. In fact, the RS232C standard appears to require this. To reduce the problems which this requirement may cause, some asynchronous cards permit the card to be configured so that certain inputs are forced true ("on"), in order to internally simulate connection to an external signal. This is done by placing shorting plugs ("jumpers") at appropriate positions on the card, or perhaps by presetting certain DIP switches. Consult your hardware documentation.

In most situations, it is preferable to simulate an external signal by jumpering selected wires in the external RS232 cabling, as is done in constructing the null modem cable mentioned in Appendix B.

For example, the AST SixPakPlus card allows CTS, DSR, and/or CD to be forced true with shorting plugs. In its factory configuration, the card expects the connected device to drive all its inputs (except Ring Indicator). Additionally, the card has been factory-configured to respond as COM1, using IRQ4; this may be modified by moving some shorting plugs. Removing certain plugs will disable the card.

Some adapters, such as IBM's asynchronous adapter, can operate in current loop interface rather than voltage interface. You'll want the voltage interface. A jumper block is attached to the card, to select the interface; it is factory set for the voltage interface. Configuration as COM1 or COM2 is also determined by a jumper block, which is factory set for operation as COM1. The IBM asynchronous adapter is described as providing an "RS232C-like interface," suggesting that it is not necessary that all of its inputs be driven.

Messy, isn't it?