PowerComm Communications Library Release Notes

Version 1.04

of March 18, 1997

RELEASE NOTES CONTENTS

1. OVERVIEW 3
2. INSTALLATION 3
3. PCL CONFIGURATION RECORDS 4
4. PCL COMMUNICATIONS COMMANDS 5
   - 4.1 PCL-CONNECT (PCL.CONNECT) 5
   - 4.2 PCL-RS (PCL.RS) 5
   - 4.3 PCL-RSDO (PCL.RSDO) 6
5. PCL UTILITY COMMANDS 6
   - 5.1 PCL-PRECOMP (PCL.PRECOMP) 6
   - 5.2 PCL-SJ (PCL.SJ) 6
6. PCL SUBROUTINES 6
   - 6.1 PCL.CP.UTY.SUB 6
   - 6.2 PCL.PORT.IO.SUB 7
   - 6.3 PCL.MODEM.IO.SUB 7
   - 6.4 PCL.RSAPI.FIO.SUB 7
   - 6.5 PCL.RSAPI.CPSR.SUB 8
   - 6.6 PCL.RSAPI.FSR.SUB 9
   - 6.7 PCL.RSAPI.TCL.SUB 9
   - 6.8 PCL.RSAPI.UTY.SUB 9
   - 6.9 PCL.RSAPI.MISC.SUB 10
   - 6.10 PCL.LOG.SUB 10
7. REMOTE SERVER 10
   - 7.1 Remote server configuration 10
   - 7.2 Terminating the remote server 11
   - 7.3 About packet timeout 12
   - 7.4 About the RS.TBL settings 12
   - 7.5 Writing your own commands 13
8. PROGRAM INFORMATION 13
   - 8.1 Conventions 13
   - 8.2 Tracing code 14
   - 8.3 Making changes 15
   - 8.4 Editing 15
   - 8.5 Installing PCL on another account 15
   - 8.6 Moving PCL to another system 15
   - 8.7 Performance issues 15
9. REFERENCE 16
   - 9.1 File contents 16
   - 9.2 Program names and descriptions 17

Version 1.04 of March 18, 1997

Terminology

Term	Description
AP	Abbreviation for Advanced Pick.
Communications port	Refers to the port that is acting as the communications pipe to a modem or to the remote computer.
Handle	A number that is used to differentiate one file/select/root variable from another through the RSAPI functions. Also used in the PowerComm user exit interface to differentiate one communications port from another for communications i/o functions.
I/O	Common abbreviation for input/output.
Local Computer	Refers to the computer that you are directly operating.
My port	Refers to the port on which you are operating the computer or running your program on.
RC	Abbreviation for Remote Computer.
Remote Computer	Refers to the computer that is on the other end of the communications link. It may or may not be another 'Pick' system.
RS	Remote Server - the program that runs on a remote Pick host computer. Also known as the Remote Server side.
RSAPI	Remote Server Applications Program Interface - the set of programs that are called by the local system to perform commands on the remote server. Also known as the Remote Server API side.

1. OVERVIEW

The PowerComm Communications Library (PCL) is a separate collection of BASIC programs layered on top of PowerComm that may be used with your code to make it easier to interface with the communications functions of PowerComm.

Pick to Pick using the Remote Server

If your intention is to communicate with another Pick system in a fully automated configuration, you should look at the remote server application program interface (RSAPI) to accomplish this. By defining some values in some control items and writing a subroutine or two, you can have a reliable method of automating Pick to Pick data communications. There are quite a few examples of using the RSAPI you can look at to get started.

Other Systems

If you are going to be communicating to another type of system, you can still take advantage of the low level communications functions that the RSAPI relies on. You will want to look at PCL.PORT.IO.SUB for handling the port i/o, PCL.MODEM.IO.SUB for a complete modem interface routine and some of the other support routines like PCL.CP.UTY.SUB for interfaces to the Start, Attach, Detach and Kill port routines.

Also, take a look at PCL.SCRIPT.SUB for common routines you may need to help automate getting 'logged on' after a successful modem connection.

Getting Started

We recommend that you first install PowerComm and get familiar with its operation before installing this Communications Library. Follow the installation steps in the next section when you are ready.

You **MUST** be running PowerComm at least version 2.2 dated June 3, 1996 or later in order to use the PCL. To find out which version of PowerComm you have, use the following command:

```
CP-VERSION (DD            (2 'D's)
```

The MFG date (if shown) should be dated June 3, 1996 or after. If no MFG date is displayed after the serial number, you will need a more recent copy of PowerComm.

2. INSTALLATION

In order to make it easier to independently load future PowerComm Software and PCL updates, you should load the library into a separate account from the PowerComm account. The PowerComm Communications Library diskette contains several T-DUMP'ed files. During the load procedure, files are created and T-LOAD'ed from diskette.

To load the PowerComm Library, follow these steps:

1. Make sure you have copied any modified programs, control files, etc. to another file or account before beginning. Or load the new PCL into a fresh account and copy in your existing control file records. **CAUTION: ALL EXISTING PCL FILES WILL BE DELETED AND RE-CREATED DURING THE INSTALLATION PROCEDURE.**

2. Make sure PowerComm is already installed in the account you want to load the Library into. You can run "Install on another account" from the PowerComm menu to do this.

3. Get logged into the target account.

4. SET-FLOPPY (AH or (BH

5. T-LOAD MD (O)

6. LOAD-PCL

At the completion of the load, you can view the release notes found in PCL.DOC.

The programs loaded into PCL.BP have not been compiled into object code. You can compile the programs just loaded by selecting the appropriate version of Pick during the LOAD-PCL install routine.

The source code contains segments of code specific to Advanced Pick (AP) and R83. As such, the source code may have to be modified (pre-compiled) through the pre-compiler before compiling into executable object code.

For Advanced Pick:

The source code is already pre-compiled for AP. To compile use the command: PCL-AP-COMPILE.

For R83:

For R83, the source code must be pre-compiled and then compiled. To do both steps automatically, use the command: PCL-R83-COMPILE.

If the PCL-R83-COMPILE command fails with:

```
[B25] Program 'LSIU.TCL.PARSE.SUB' has not been cataloged.
```

you have not performed step 2.

3. PCL CONFIGURATION RECORDS

Control items in PCL.CONTROL

There are several items in the PCL.CONTROL file.

CONFIG.TBL

This record contains the main configuration parameters for the PowerComm Library programs. It contains parameters for timeout, packet sizes and modem names defined for each port. Brief documentation is provided within the supplied item and more detailed documentation may be found in PCL.BP INC.CONFIG.TBL.EQU.

In order for PCL-CONNECT to operate, a list of port numbers in a multi-valued list along with another attribute of multi-values for the modem names (if not a direct connection) must be set up in this item.

To help differentiate the types of items in the PCL.CONTROL file, a naming convention is followed but not enforced for the remaining items.

MODEM*name

Modem definitions are stored in this item. A standard modem definition is supplied named MODEM*STD. You can copy and customize as needed for other modem types you might have connected to the system. Brief documentation is provided within the supplied item and more detailed documentation may be found in PCL.BP INC.MODEM.TBL.EQU.

LOG*STARTUP & LOG*WRAPUP

These unstructured records contains a list of logging commands to execute when the PCL-CONNECT program starts (or stops) processing a remote computer (RC). You can name the specific LOG records to use in the PCL.RC item. Documentation on the types of logging commands can be found within the examples as well as PCL.BP PCL.LOG.SUB.

SCRIPT*LOGON & SCRIPT*LOGOFF

These unstructured records contain a list of script commands that can be used to get logged on (or off) a remote computer. You can name the specific SCRIPT records to use in the PCL.RC item. Documentation on the types of script commands can be found within the examples as well as PCL.BP PCL.SCRIPT.SUB.

<u>Remote computer (RC) items in PCL.RC</u>

This file contains one item per remote computer definition. It contains fields like port numbers to use, phone number to dial, number of retries to connect, communications port start parameters, etc. Documentation for each of the attributes can be found in PCL.BP INC.RC.TBL.EQU. To avoid redundant entries, you can use the 'DEFAULT' record in this file to supply parameters for any NULL attribute in RC items. To avoid using the default and keep a NULL parameter, simply use "\" as the first character of the field. By using the 'DEFAULT' record, you can maintain most of the parameters for all of the RC's in one place and supercede them as needed within the RC items.

4. PCL COMMUNICATIONS COMMANDS

The three mainline communications related programs supplied and their descriptions are described below. Further documentation may be found at the top of the program source code for each of these programs. The source code documents the TCL command line interface and all of the available commands and options. Some examples using these programs are included later in this document.

A proc front-end for each of these programs is defined in PCL.PROC that allows you to run the program names (shown in parenthesis) from the TCL command line with parameters.

4.1 PCL-CONNECT (PCL.CONNECT)

This mainline routine is intended to be used within a program you write for automating connections/transfers to remote computers. Configuration information found in the PCL.RC file and the PCL.CONTROL file are used to start the appropriate communications port, and connect to the remote computer.

If the remote computer is via a modem, it will use a modem control item (MODEM*name) to dial the phone number, and wait for connect. It then runs a logon script (SCRIPT*name) to get logged on. Upon successful logon, a command line supplied program is run which can perform whatever communications task is required. Upon return from the subroutine, the remote computer is logged off via a user-defined script, disconnected, and the port killed.

If you are connecting to a remote Pick computer then you can run the PCL-RS program on the remote system to perform functions under program control from the local computer. Your subroutine can do whatever it needs to do to move files, perform TCL commands, etc. A subroutine example (SAMPLE.TRANSFER.SUB) is provided to help get you started.

A sample driver program SAMPLE.PCL.CONNECT.DRIVER is provided to give you an idea of how to call the PCL-CONNECT routine from another basic program.

Full documentation on PCL-CONNECT can be found in PCL.BP PCL.CONNECT.

4.2 PCL-RS (PCL.RS)

This mainline routine, the Remote Server (RS), is the program that is run on the **remote** Pick computer. Briefly, the RS simply waits for a command packet, performs the command, returns the results and wait for the next command packet. There are over 35 commands it can process through the Remote Server API (RSAPI). These functions may be initiated through subroutine calls on the local machine to the PCL.RSAPI.xxx.SUB programs.

The PCL-RS proc is called from the PCL-RS-LOGON proc both found in the PCL.PROC file.

See the main section in this document 'REMOTE SERVER' for more detailed information.

4.3 PCL-RSDO (PCL.RSDO)

This mainline routine, run on the **local** system, provides a TCL command line interface to 'DO' some of the RSAPI routines. You can use this program to GET and SEND files to the remote server and CALL your own subroutines for testing. The remote computer must be connected and at TCL or already running the remote server program 'PCL.RS'.

You must be connected to a remote computer and either at TCL or in the remote server program 'PCL-RS' on the remote before using the 'PCL-RSDO' command on the local system.

You can test the link to the remote server program running on the remote computer by using the PCL-RSDO program and the 'PING' command:

```
PCL-RSDO PING 3 1000 1000 C
```

The 'C' option will calculate a minimum timeout value for the baud rate, and physical block sizes in use. This is a best case timeout value and a processing delay should be added to the value shown.

Other examples can be found in the PCL.RSDO program in PCL.BP.

5. PCL UTILITY COMMANDS

The following 2 mainline programs are not intrinsic to the communications aspects of the PowerComm Library but are provided as utility programs.

5.1 PCL-PRECOMP (PCL.PRECOMP)

This mainline utility function is used to pre-compile the PowerComm Library source code for a different Pick version. It is used in the PCL-R83-COMPILE proc found in PCL.PROC.

5.2 PCL-SJ (PCL.SJ)

This mainline utility routine may be used to split large items into several parts before transmission and joining the small items back together again after transmission. This may be needed with R83 systems where you may run into problems with not enough workspace problems.

6. PCL SUBROUTINES

If you use any of these programs you will need to first call the PCL.INIT.VARS.SUB program to initialize the global variables and load the user mode definitions. See the mainline programs for examples.

6.1 PCL.CP.UTY.SUB

This program is responsible for interfacing to the PowerComm TCL commands for starting, attaching, detaching and killing a communications port. It also contains the routine for getting a communications port handle which is used by the PCL.PORT.IO.SUB functions for communications port i/o.

This subroutine provides the following communications functions:

StartPort, AttPort, GetHandle, DetPort, KillPort

Examples of interfacing to this subroutine can be found in the program PCL.RC.COMM.SUB.

6.2 PCL.PORT.IO.SUB

This program is responsible for handling all communications to the active communications port, whether it is a PowerComm communications port or the local port.

This subroutine provides the following communications functions:

> SendString, ViewSendString, SendData, GetRecvData, GetRecvDataCount, GetRecvDataChar, GetRecvDataCharTimed, ViewRecvData, ClearRecvBuffer, GetPortInputCount, GetPortOutputCount, WaitForData, ViewWaitForData, WaitQuiet, ViewWaitQuiet

A full description of function each can be found within the PCL.PORT.IO.SUB program. Several examples using these communications functions can be found in PCL.MODEM.IO.SUB, PCL.PK.IO.SUB and PCL.SCRIPT.SUB.

6.3 PCL.MODEM.IO.SUB

You can also take advantage of a fully functional modem interface program called PCL.MODEM.IO.SUB which along with a control item, 'MODEM*name' eliminates the drudgery of getting all the modem communications timing correct all under table-driven control.

It provides the following modem functions:

> Reset, Attention, Init, ResultsOn, ResultsOff, Hangup, CommandMode, Dial, WaitForDialResult, UserString, CancelDial, GetInfo.

For examples on how to use PCL.MODEM.IO.SUB, refer to the program PCL.RC.COMM.SUB which includes the Connect and Disconnect routines for communicating to a remote computer.

A simple program called SAMPLE.MODEM.IO demonstrates several parts of the PCL. This example uses port 1:

```
CP-START 1,19200 (A
SAMPLE.MODEM.IO
```

This program logs the result of the "AT I6" command (viewed on the screen) to the log file and then prompts you to edit the log item created.

6.4 PCL.RSAPI.FIO.SUB

The remote server file i/o (FIO) category of RSAPI functions includes those functions that you would normally use on local files such as OPEN, READ, WRITE, DELETE, SELECT, READNEXT, ROOT, and KEY. The PCL RSAPI program PCL.RSAPI.FIO.SUB provides similar functionality but over the communications port. For example, instead of a READ statement, a subroutine is called passing to it similar parameters found in the standard read statement.

The following are functions available in this program:

> OpenFile, CloseFile, CloseSelect, CloseRoot, Clearfile, ReleaseAll, ReleaseID, Select, SelectTCL, SelectItem, ReadnextID, ReadnextItem, ReadnextItemU, Read, ReadV, ReadU, ReadVU, Write, WriteV, WriteU, WriteVU, Delete, Root, Key.

In order to use a file on the remote system you must first open the file on the remote system. If successful, one of the return parameters is a number (or a handle) to the file on the remote system. When using a subsequent file i/o function, like Read or Write, you must supply to the RSAPI subroutine this file handle. Similarly, when using a select list with ReadnextID or ReadnextItem, you also pass the select variable handle to the RSAPI function.

If you use the btrees on AP, you must first open the btree root index with the Root function which returns a root handle that is used with subsequent Key functions.

Each of these FIO functions can be found in the PCL.RSAPI.FIO.SUB program along with a description of input and output parameters and an example you can copy into your code.

A working example using several of these functions can be found in the test program called TEST.RSAPI.FIO.SUB. Within this program there are 4 techniques of performing the identical functionality demonstrated as options 1 through 4. This program can be demonstrated with the following command:

```
PCL-RSDO CALL TEST.RSAPI.FIO.SUB     (options 2,3 and 4)
```

The preceding examples use files called 'LOCAL.DATA' on the local system and 'REMOTE.DATA' on the remote system. You can create and populate the local file yourself, or you can use option 6 in the test program to create these local and remote files for you along with 20 small 500 byte items. The following command will do this for you:

```
PCL-RSDO CALL "TEST.RSAPI.FIO.SUB 6"
```

For speed, there is a combined function of ReadnextID and Read called ReadnextItem. This is demonstrated as option 3 in the test program. This 'ReadnextItem' function reads the next item from a select list on the remote, and if not end of file condition, also reads and transmits the item body too. This can save the overhead of 2 extra packets going back and forth when your intention is to always read the next item regardless of the item ID. However, if you are transferring several items, you should take advantage of the speed with the 'SendItems' and 'RecvItems' functions described in the PCL.RSAPI.FSR.SUB program.

6.5 PCL.RSAPI.CPSR.SUB

The following are functions available in this program:

SendFile, RecvFile, SendSpool, RecvSpool

This category of functions provide an interface to the PowerComm commands: CP-SEND, CP-RECV, CP-SEND-SPOOL, CP-RECV-SPOOL. The SendFile and RecvFile functions can be used when you need to transfer multiple items at a time and you need the extra speed.

For moving spooler entries, you can use SendSpool and RecvSpool.

The test program TEST.RSAPI.FIO.SUB demonstrates several file i/o functions. It also shows there are several ways of moving items between systems.

```
PCL-RSDO CALL "TEST.RSAPI.FIO.SUB 1"     (option 1)
```

Option 1 in this program calls the PowerComm CP-SEND and CP-RECV programs through the program PCL.RSAPI.CPSR.SUB. The following discusses the advantages and disadvantages of using PCL.RSAPI.CPSR.SUB:

<u>Advantages of using PCL.RSAPI.CPSR.SUB</u>

Can be about the fastest method to move items between systems.

Can handle large 32k items even on R83 because of how CP-SEND and CP-RECV are implemented. You will not run out workspace with these commands because they are called from an execute statement.

You can take advantage of the automatic block size leveling protocol features.

<u>Disadvantages of using PCL.RSAPI.CPSR.SUB</u>

Invalid parameters to the command will not be reported in any easy to decipher format. You would have to parse through remote system output to discern one type of error from another. For instance, a bad file name specified will cause the Pick error message '[10] File name is missing' or '[201] 'xxx' is not a file name'.

An aborted transfer is more difficult to synchronize with because the packet protocol is not compatible with the PCL and it is also running at different execute level.

6.6 PCL.RSAPI.FSR.SUB

The following are functions available in this program:

SendItems, RecvItems

This category of functions provide a fast way of moving multiple items at a time. Similar to SendFile and RecvFile using CP-SEND and CP-RECV, these 2 functions work together with the FIO functions in that you pass to them file and select handles or file and select variables.

The commands 'GET' and 'SEND' in the PCL.RSDO program are good examples of using these 2 functions.

A working example using several of SendItems and RecvItems can be found in the test program called TEST.RSAPI.FIO.SUB. This program can be demonstrated with the following command:

```
PCL-RSDO CALL "TEST.RSAPI.FIO.SUB 2"     (option 2)
```

Option 2 performs the same end result as option 1 but uses the program PCL.RSAPI.FSR.SUB. It is similar in the aspect that multiple items are automatically blocked and deblocked into a fixed large size logical packet for transmission. However, on R83 systems with larger items (>5K), it uses a slower technique to write the item to the file to avoid workspace problems. It also does not include any block size adjusting capabilities but is very fast and simple and provides the ability to track the last item ID successfully written as well as transmission statistics.

This is the recommended program to use for multiple item transfers if workspace is not an issue with large items. In addition, if you want to add your own features, you can always copy and modify the source code to suit your needs.

6.7 PCL.RSAPI.TCL.SUB

The following are functions available in this program:

LocalTCL, RemoteTCL

This subroutine contains functions for performing TCL commands locally and on the remote server.

Several options are available for executing TCL commands on the remote server. You can use the CAPTURING clause on the remote server and discard or send the results. You can save the VIEW'ed output and translate CRLF's to attribute marks. You can send the command to the remote server, and quit the remote server after the command completes.

For example, you may want to send files back and forth, and as the last step, send a TCL command to the remote system that posts the transferred files to their destinations. Once this program is off and running, you may want to hang up and move on to the next remote system saving connect time charges.

A word of caution when running programs when the modem is not online. You could possibly lock up the modem if your program generates output that contains "AT" in it. You can minimize or eliminate the output in the posting program or use the execute CAPTURING clause. Beware that the capturing clause could cause workspace problems on R83 if the size of the variable captured is large enough. You can also turn off command recognition on the remote system's modem if you never use it for dialing out.

A working example using several of these functions can be found in the test program called TEST.RSAPI.FIO.SUB. This program can be demonstrated with the following command:

```
PCL-RSDO CALL "TEST.RSAPI.FIO.SUB 1"     (or options 5,6 and 7)
```

6.8 PCL.RSAPI.UTY.SUB

The following are functions available in this program:

ClearStack, ChangeTimeout, ChangeRecvBufferSize

These functions do not cause data communications, they are used to change the timeout, or recv buffer sizes and restore the prior value with a function call.

An example using several of these functions can be found in the program called PCL.RSDO.

6.9 PCL.RSAPI.MISC.SUB

The following are functions available in this program:

GetFVList, RemoteLog, Ping, GetOptions, SetOptions, System, Call

The GetFVList is used mainly for debugging as a way to find out what the remote server currently has open in way of file/select/root variables.

The RemoteLog function provides a way to send log commands to the remote server and act on them.

The Ping function is used by PCL.RSDO for testing and reporting of link speed throughput.

The GetOptions and SetOptions functions provide an interface to query and set certain remote server options.

The System function provides an interface to SYSTEM() on the remote server.

The Call function provides an interface to a subroutine call on the remote server.

A working example using several of these functions can be found in the test program called TEST.RSAPI.FIO.SUB. This program can be demonstrated with the following command:

```
PCL-RSDO CALL "TEST.RSAPI.MISC.SUB"
```

6.10 PCL.LOG.SUB

Another useful program PCL.LOG.SUB can be used for helping to debug, or simply to log the results of data communications activities. Many examples can be found since it is used in almost all of the PCL software.

7. REMOTE SERVER

The remote server interface is layered on top of the low level PCL functions. The PCL.PK.IO.SUB program is called from both ends of the communications link; the remote server API side and the remote server command processing side. It is responsible for packeting data into logical blocks and sending them to the remote system. It is also responsible for getting data from the remote computer and validating the data according to the computed length, checksum and packet sequence numbering.

It calls the PCL.PORT.IO.SUB program to handle the actual communications port i/o regardless of context. It doesn't care if the i/o is local (in Me mode) or over the PowerComm communications port. The PCL.PORT.IO.SUB program decides which user mode communications functions to call according to the context.

7.1 Remote server configuration

The example script supplied in the PCL.CONTROL SCRIPT*LOGON item to get logged on to a remote Pick computer attempts to logon to an account (or user) named 'PCL-RS-LOGON'.

Make sure you have created this logon name (or on AP a user item in the users file), that logs on and runs the PCL-RS-LOGON proc in the MD.

```
ED SYSTEM PCL-RS-LOGON          fill in     (or MDS for AP)
ED USERS PCL-RS-LOGON           fill in     (for AP)
```

This account (in SYSTEM or MDS) would normally be a "Q" logon account type that references the account you are going to have the remote server program run in.

Here is an example setup on Advanced Pick:

```
    MDS PCL-RS-LOGON
001 Q
002 DEV.CP                  <- your account name here
003
004
005
006
007
008 SYS2
009 L
010 10

    USERS PCL-RS-LOGON
001 remote server user logon id
002
003
004
005
006
007
008 sys2
009 r
010
011
012 term-type
013 sp-assign
014 set-sym gsym
015 speller-off
016 time-date-off
017 tcl-hdr-off
018 brk-debug
019 esc-data
     (your 'user logon' commands may vary)
020 LOGTO PCL-RS-LOGON
```

In the account MD of 'DEV.CP':

```
    MD PCL-RS-LOGON
001 PQ
002 (PCL.PROC PCL-RS-LOGON
```

This proc in turn runs the 'PCL-RS' program and logs off when done. Any customization of flow control and other settings should be done in the PCL.PROC PCL-RS item.

7.2 Terminating the remote server

If you need to terminate the remote server program while in direct mode (CP-DIR) you can type 'END' or 'OFF'. You may need to wait up to 2 seconds before typing 'END' or 'OFF'. In other words, pressing <ENTER> 'END' <ENTER> quickly in succession may not terminate the server. In fact, if you press enter and you are in the remote server, you will get the last packet the remote server tried to send. Resist the temptation 'to see where you are' and just type 'END' <ENTER>.

The following entries are special to the remote server packet input program:

END, STOP	Terminates the server with a BASIC "STOP" statement. The PCL-RS proc will continue.
OFF, EXIT	Terminates the server with a BASIC "CHAIN 'OFF'" statement.
ABORT	Terminates the server with a BASIC "ABORT" statement. The PCL-RS proc will not continue.
_DEBUG	Causes the server to flush the log and enter the debugger. Use 'G' <ENTER> to resume when ready.
PCL-RS	Restarts the server and re-initializes all variables.
PCL-RS V	Causes the remote server to display <rs-ready>

7.3 About packet timeout

The packet timeout you select should be sufficient to allow a physical maximum size packet to be transmitted and a physical maximum size acknowledgment received. For example, if you are running at 9600 baud using a packet size of 5000 bytes, a 12 second packet timeout is not sufficient.

You will achieve maybe 90% of actual baud rate or around 864 characters per second which means about 6 seconds to transmit 5000 bytes, allow for some turnaround processing time and another 6 seconds to fully receive 5000 bytes. A minimum of 12 seconds is required in this case. Add to this value a processing delay to arrive at the final packet timeout value to use.

You can calculate the timeout using this formula:

size / (baud.rate / 10) * 90% + (processing.seconds) = Timeout

Where 'size' is the combined maximum input and maximum output data lengths.

For example, using a input size of 5000 and a output size of 5000 with a baud rate of 9600 and a processing delay of 5 seconds the calculated timeout value would be:

10000 / (9600 / 10) * .9 + 5 = 16.6 seconds

The magic number 90% is probably a best case value to calculate actual throughput for a given baud rate. Your mileage may vary. The magic number '10' takes into account 8 data bits plus start and stop bits.

The default of 12 seconds is valid for a block size of 3500 bytes at 19200 baud using 8 seconds of processing delay. You will need to increase the 12 seconds to a higher value if you choose to use a larger block size or a slower baud rate or a slower remote server host.

Another point to consider is your actual throughput with a 14,400 modem vs a true 19,200 direct connection. You may need to increase the timeout value slightly higher.

You can use the PING command with the 'C' option to calculate this minimum value:

```
PCL-RSDO PING 1 3000 3000 C
```

7.4 About the RS.TBL settings

The packet timeout and packet transmit length is driven by the remote server program. The remote server (on the remote computer) looks at this item and tells the local machine how long to wait between packets 'PacketTimeout' and how much data it can receive at once 'InputDataLength'.

This is so a particularly slow computer can have a longer packet timeout and a smaller packet size and the local system will automatically adjust its timeout and transmit packet size when it connects and exchanges the 'init' packets.

7.5 Writing your own commands

The remote server program (PCL.RS) can also handle your own user-defined custom commands. An example of a custom command can be found in the test program TEST.RSAPI.USER.SUB.

8. PROGRAM INFORMATION

8.1 Conventions

The following are conventions used in the PowerComm library.

<u>Variables</u>

The prefixed variables describe the content and/or the use of the variable.

Example	Description
fnControl	file name variable
Example:	OPEN fnControl TO fvControl ELSE
fvControl	file variable
svControl	select variable
rvControl	root variable (AP btrees)
Control$	item variable
Example:	READ Control$ FROM fvControl, idControl ELSE
Control$Description	item attribute variable
Example:	READV Control$Description FROM fvControl, idControl ELSE
Control$$Description	equated attribute#
Example:	READ Control$ FROM fvControl, idControl ELSE Desc = Control$<Control$$Description>
idControl	item id variable idFile
Example:	READV Control$Description FROM fvControl, idControl ELSE
bView	boolean flag value is zero '0' or not zero
Example:	IF bView THEN GOSUB ShowData
iString	'index' variable index into string
nString	'max of' variable as in leNgth of string
Example:	FOR iString = 1 TO nString
pString	'position of' variable
Example:	pString = INDEX(Item$, SearchString, 3)

uReadPort	user mode even though user modes in this library are 'global', they are not prefixed with G$

Example: Result = OCONV(Handle, uReadPort)

G$PortNo	global variable 'PortNo' all global variables in PowerComm Library are simply prefixed with G$ (except for user exits)

Example: G$RC.Description
G$bViewFilter
G$bLogActive

GU$UserVar1	your own user global variables, as suggested in PCL.BP.USER INC.PCL.COM.USER

<u>Constants</u>

Constants are uppercase.

Example	Description
AM	Equated attribute mark, CHAR(254)
VM	Value mark, CHAR(253)
SVM	SubValue mark, CHAR(252)
SP	Single space, " "
SECONDS.PER.DAY	Number of seconds per day - not changing anytime soon

Example: Time = DATE() * SECONDS.PER.DAY + TIME()
nItem = DCOUNT(Item$, AM)

Program functions are program name uppercase followed by the function in mixed case. 'SUB' is omitted.

Example	Description
PCL.CP.UTY.StartPort	Start port function in PCL.CP.UTY.SUB

Example: CALL PCL.CP.UTY.SUB(PCL.CP.UTY.StartPort, ...)

8.2 Tracing code

The global option tilde "~" may be used for debugging and understanding the layers of PCL. Refer to the program PCL.DEBUG.SUB for options. To see an example trace on the screen, try:

```
PCL-RSDO PING 5 3500 3500 C (~
    -or-
PCL-CONNECT PORT-2 (~
```

Then, sometime during program execution, press 'E' to view current stats.

The pre-compiler (PCL.BP PCL.PRECOMP) can be used to remove trace flag testing and for your own debugging assistance.

To comment out the trace flag testing code in the PowerComm Library use the following command replacing 'AP' below for 'R83' if running on R83:

```
PCL-PRECOMP PCL.BP * (F,O) {AP} {TRACE} {C}
```

8.3 Making changes

If you need to make changes to the PCL.BP programs, do not modify programs in the PCL.BP file. Instead, copy programs from the PCL.BP file to the PCL.BP.USER file and catalog the PCL.BP.USER version.

To add common variables, add them to PCL.BP.USER INC.PCL.COM.USER.

8.4 Editing

We recommend using a full screen editor to maintain and modify the programs and the control file items. The Pick line editor (ED) wrapping is really ugly in 80 columns. A little improvement can be gained by working in 132 columns and adjusting your term settings accordingly.

Some other editor options:

Jet - You can contact VMark for current releases available on most all AP platforms. Although not a good 'Word Processor', the JET-EDIT mode of this program is fast and while not the ultimate editor, this is what we used for EDITING most of the PCL code.

Pegasus - From Weaver Consulting. A relatively new offering and we know nothing about it.

RED - A very large Pick shareware editor (several megabytes) but if you've got the time to learn it, it does it all.

Update (U) - Comes with AP but not well suited for editing code since it word wraps the code at the end of the line instead of scrolling left and right on the screen.

8.5 Installing PCL on another account

To install PCL to another account, you will need to perform the following steps:

1. Create Q pointers to the account containing the PCL files; PCL.BP, PCL.BP.USER, PCL.CONTROL, PCL.LOG, PCL.MD, PCL.PROC, PCL.RC etc.

2. Copy the PCL.MD file items to the target master dictionary (MD) file.

3. Catalog the PCL.BP programs and any other files you may have created containing subroutines, such as the PCL.BP.USER file.

8.6 Moving PCL to another system

For convenience, you will find a package template in the PCL.PACKAGE file called 'PACKAGE' that contains all of the commands you will need to move all of the original PCL files to another system using PowerComm commands. Notes and instructions are within the package command item.

Look at PCL.BP PCL.PRECOMP to see how to remove comments from code if you want to save transmission time.

8.7 Performance issues

<u>Logging</u>

Throughout the PCL are various tests for logging options. These tests are very fast. However, the actual logging when the flags are true may slow down the overall performance of the PCL functions. The benefits for debugging and tracking purposes may outweigh the slight slow down. You may want to experiment with a run using the log functions and not using the log functions to compare the performance.

Some performance can be gained by not logging the following event types:

Event description	Log option to use
Packet I/O	PK-IO:OFF
Send/Recv/View data	SD:OFF RD:OFF VD:OFF
Limiting event size	MAX-EVENT-SIZE:50 (or a small number)

But, if you log packet i/o, use:

PK-IO:TRIM;40 (or a small number)

And, if you log remote server functions use:

RS:TRIM;40 (or a small number)

During 'SendItems' and 'RecvItems', there is only 1 'RS' entry made, therefore you do not pay a performance hit for each item, unless of course you are logging packet i/o also.

<u>Packet Sizes</u>

Select the largest packet size you can (up to 9019) bytes to minimize the turnaround time. You can use the PING command with PCL-RSDO to test the reliability of a larger packet size. Do not forget to adjust the packet timeout values as needed.

The packet data input/output lengths must not exceed the length of the CP-START input/output buffers or packet truncation will occur.

9. REFERENCE

9.1 File contents

File name	Contents
PCL.BP	Main PCL basic program source code file. All of the subroutines should be cataloged in this file.
PCL.BP.USER	PCL basic program user source code file. This is where you might put your code. This file initially contains a couple of items that are automatically 'INCLUDE'd as part of the source code in PCL.BP. This provides you with a hook to add some additional global variables to the initialization routine in PCL.INIT.VARS.SUB.
PCL.CONTROL	This file contains the control items that are used by the PowerComm Library programs.
PCL.DOC	This file contains a flow map of the PCL.BP file.
PCL.LOG	This file can be used to log the results of using the PCL commands. Initially empty.
PCL.MD	This file is copied to the MD and contains short procs that branch to the real procs in PCL.PROC. This is so you can easily install PCL to other accounts and still make changes to the PCL.PROC file in one place.
PCL.PACKAGE	This file contains a CP-PACKAGE command template 'PACKAGE' that is helpful for moving the PCL files to another system if needed.
PCL.PROC	This file contains several front-end procs to the PCL.BP programs.

PCL.RC	This file contains the remote computer (RC) definition items that are used when using PCL-CONNECT..

9.2 Program names and descriptions

The following are include items 'INCLUDE'd by programs.

Include name	Description
INC.ASCII.EQU	Standard equate items for ASCII control chars.
INC.CONFIG.TBL.EQU	'CONFIG.TBL' attributes and documentation.
INC.MODEM.TBL.EQU	'MODEM*name' attributes and documentation.
INC.PCL.COM	Defines PCL program common (global variables).
INC.PCL.EQU	PowerComm Library equates (part 1 of 2).
INC.PCL.EQU.PART2	PowerComm Library equates (part 2 of 2).
INC.PORT.IO.SUBS	Port i/o subs for PCL.PORT.IO.SUB.
INC.RC.TBL.EQU	'PCL.RC item' attributes and documentation.
INC.RSAPI.SUBS	Common subs for PCL.RSAPI.xxx programs.
INC.STD.EQU	Standard equate items (non PowerComm Library related).

The following are PCL subroutines that may be called by the PCL mainline routines or directly from your programs.

Subroutine name	Description
PCL.CONFIG.SUB	PCL.CONTROL 'CONFIG.TBL' interface routines.
PCL.CP.UTY.SUB	PowerComm 'CP' command interfaces for starting, attaching, detaching, killing a port. Also interface for getting communications port handle.
PCL.DEBUG.SUB	Debug utility
PCL.INIT.VARS.SUB	Program to initialize PCL global variables.
PCL.INPUT.SUB	Subroutines for handling input.
PCL.LOG.SUB	Log file logging routines.
PCL.MODEM.IO.SUB	Modem i/o routines for handling all modem programming.
PCL.MODEM.TBL.SUB	PCL.CONTROL 'MODEM*name' interface routines.
PCL.MSG.SUB	Screen message display routines.
PCL.PARSE.LINE.SUB	Parses command line into separate components.
PCL.PK.IO.NOTES	Notes for PCL.PK.IO.SUB.
PCL.PK.IO.SUB	Packet i/o routines for sending and receiving data packets to/from the remote server.

PCL.PK.UTY.SUB	Packet utility routines.
PCL.PORT.IO.SUB	Communications port (or local port) i/o routines.
PCL.PORT.TBL.SUB	PCL.CONTROL 'PORT.TBL' interface routines.
PCL.RC.COMM.SUB	Remote computer communications routines for starting a remote computer, connecting, logging on, logging off, disconnecting, killing a port. (calls PCL.CP.UTY.SUB).
PCL.RC.TBL.SUB	PCL.RC file interface routines.
PCL.RSDO.NOTES	Documentation notes for PCL.RSDO.
PCL.RS.UTY.SUB	Remote server utility routines; init packet exchange, verify server, etc.

The following 'RSAPI' programs may be called by your programs to perform various tasks on the connected remote computer.

PCL.RSAPI.CPSR.SUB	RSAPI for CP-SEND/RECV, CP-SEND-SPOOL/CP-SPOOL-RECV.
PCL.RSAPI.FIO.SUB	RSAPI for file i/o routines; open, read, write, delete, readnext id, select, etc.
PCL.RSAPI.FSR.NOTES	Notes for PCL.RSAPI.FSR.SUB.
PCL.RSAPI.FSR.SUB	RSAPI routines for sending and receiving multiple items at a time in blocked mode for speed.
PCL.RSAPI.MISC.SUB	RSAPI miscellaneous routines.
PCL.RSAPI.TCL.SUB	RSAPI routine for performing TCL commands on the remote computer.
PCL.RSAPI.UTY.SUB	RSAPI utility routines for changing/restoring timeouts, recv buffer sizes.

The following remote server 'RSCMD' programs run on the remote server side of the communications link to perform the commands initiated by the corresponding 'RSAPI' named programs.

PCL.RSCMD.CPSR.SUB	Remote Server side command processing complement to PCL.RSAPI.CPSR.SUB.
PCL.RSCMD.FIO.SUB	Remote Server side command processing complement to PCL.RSAPI.FIO.SUB.
PCL.RSCMD.MISC.SUB	Remote Server side command processing complement to PCL.RSAPI.MISC.SUB.
PCL.RSCMD.TCL.SUB	Remote Server side command processing complement to PCL.RSAPI.TCL.SUB.

The following are more PCL subroutines.

PCL.SCRIPT.SUB	Script processing routine for the 'Logon' and 'Logoff' processing in PCL.RC.COMM.SUB.
PCL.STRING.SUB	String processing subroutine functions.

Instead of using a script to get logged on, you can look at the following example program to see how you might write your own logon program for more difficult logons than the script routines can handle.

SAMPLE.LOGON.SUB

The following is a stand alone 'TEST' program that demonstrates several PCL functions that do not use any remote server programs.

SAMPLE.MODEM.IO

The following 'TEST' programs may be used with 'PCL-RSDO' to demonstrate the remote server API functions. See the PCL-RSDO examples in this document.

TEST.RSAPI.FIO.SUB
TEST.RSAPI.MISC.SUB
TEST.RSAPI.TCL.SUB
TEST.RSAPI.USER.SUB

The following are 'TEST' programs run on the remote server side and invoked indirectly through the remote server as examples in 'TEST.RSAPI.MISC.SUB' and 'TEST.RSAPI.USER.SUB'.

TEST.RSCMD.MISC.CALL.SUB
TEST.RSCMD.USER.SUB

For further information

Print out the PCL.BP items INC.PCL.COM, INC.PCL.EQU and INC.PCL.EQU.PART2 for descriptions of all variables and constants used in the PowerComm Library software.

In addition, you can look at the item 'FLOW.MAP' in PCL.DOC to see program flow by program name.