The GCOM PAD
User Guide

(Packet Assembler Disassembler)

Version 1.0

September 12, 2002

Table of Contents

Introduction

The GCOM PAD system consists of two user level programs. `Gcom_hpad` is a daemon that is run on the "host" computer. `Gcom_tpad` is an interactive program that is run on the "terminal" computer.

The acronym PAD stands for Packet Assembler Disassembler. It is a term coined by the CCITT to describe what amounts to an X.25 based terminal concentrator. The CCITT model for a PAD consists of the following:

- It is assumed to be a stand-alone unit that performs only the specified functions.
- It contains asynchronous communication ports to which terminals may be attached directly or via modem.
- The async ports have the ability to control and sense the signals DTR, RTS, DSR, DCD, CTS, and RI.
- It contains a synchronous port over which communications follows the X.25 protocol.
- The PAD functions utilize the procedures specified in X.28 to communicate with the async terminals (referred to as "start-stop DTE" in the X.28 standard).
- The PAD functions utilize the procedures specified in X.29 to communicate with the remote X.25 DTE (which we refer to as the "host" computer).
- The PAD functions utilize the parameters specified in X.3 to control the behavior of the async terminals.
- Each async terminal's byte stream into and out of the PAD is mapped to one X.25 virtual circuit to the host computer.

The function of the host end of the PAD connection is implied by the CCITT standards. It is not specified by these standards.

The GCOM PAD software implements the functions of the CCITT model using two programs, `Gcom_hpad` and `Gcom_tpad`. The `Gcom_hpad` program performs the functions of the "host" end of a PAD connection. The `Gcom_tpad` program performs the functions of the "terminal" end of the PAD connection. These programs are described in their individual sections below.

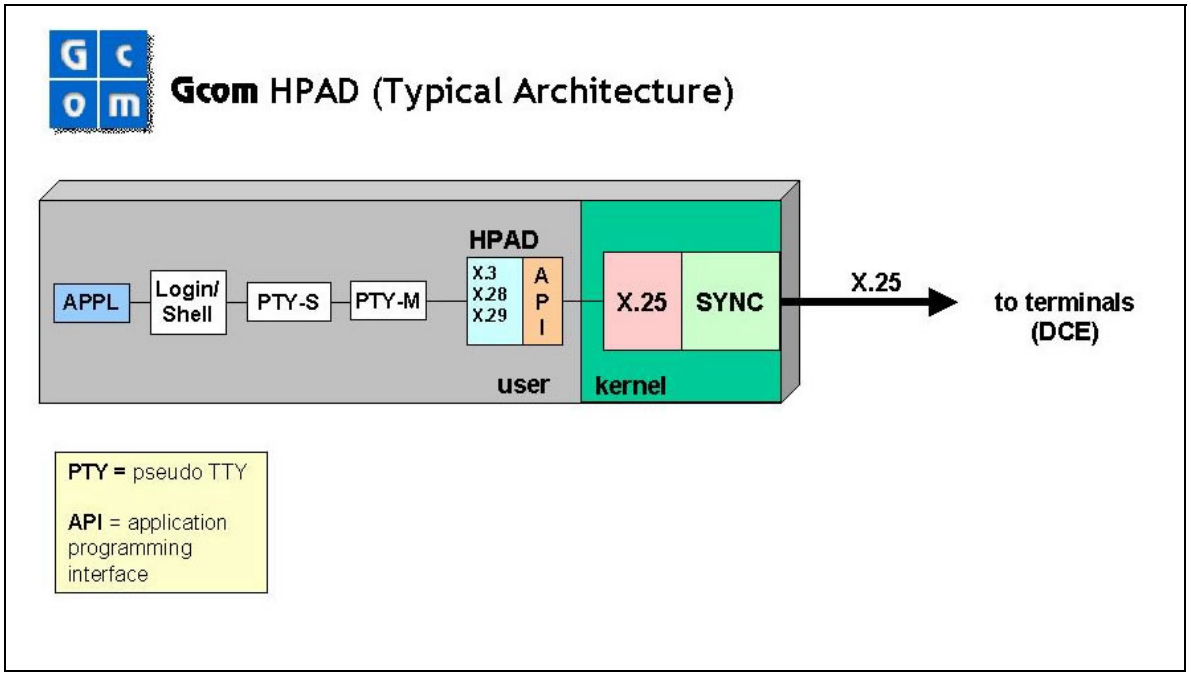
Gcom_hpad

The `Gcom_hpad` (host PAD) program is run on a system to provide the "host" end of the PAD connection. This program runs as a daemon process, typically in the background, listens for incoming X.25 calls and forks child processes to manage each X.25 virtual circuit.

Each child process operates a pseudo-tty port as the master and an X.25 virtual circuit to the remote terminal PAD. By default the slave side of the pseudo-tty port runs the "login" program, which allows the remote user to log in to the system and obtain a shell session.

Any number of copies of the `Gcom_hpad` daemon can be run to listen for different types of incoming X.25 calls.

The `Gcom_tpad` and `Gcom_hpad` can both be operating at the same time and will share the virtual circuits of the X.25 access lines.



Invocation

The `Gcom_hpad` is invoked using the following command line syntax:

`Gcom_hpad` *options*

The values for *options* are:

–A Port-Number	The default for this parameter is 0. If set to a nonzero value, the *Port-Number* is taken to be an NPI Lower Point of Attachment (LPA) number and the listening for incoming calls is restricted to that port. The LPA is generally equivalent to physical line number. The default value of 0 means listen for calls from any line.

–B	Run in background mode. This is the preferred mode for daemon operation.
–d Debug-Flags	Debug flags for Gcom_hpad. Default value is 0x0001. Values are internal and should be used at the direction of GCOM Support.
–l Listen-Address	(Lower-case "L"). This is the X.121 address on which to listen for incoming X.25 calls. It can contain the wildcard characters "*" and "?" which perform the same function as they do for UNIX shell file name expansion. The default address of "*" listens for all incoming calls arriving at this host.
–L Log-Name	Sets the name of the log file created by Gcom_hpad. The default name is Gcom_hpad.log. ***Note*:** When running multiple copies of Gcom_hpad you should name the log files differently for the different copies of the program.
–O Log-Options	NPI log options. These are bit-encoded. They are described in the *NPI API Guide* and in the file <gcom/npiapi.h>. The default value is **0x0013**.
–p PAD Parameters	Sets the values for the X.3 parameters that are sent to the remote (terminal) PAD after the X.25 call is accepted by Gcom_hpad. The notation is a comma-separated list of pairs of numbers. The syntax of each pair is *parameter:value*. The *parameter* specifies the X.3 parameter reference number. The *value* specifies the value to use for that parameter. Example: –p 1:0,3:0,4:1
–P	Dump PAD parameters to the log file.
–r Remote-Address	This parameter causes Gcom_hpad to listen for incoming calls only from the remote DTE whose X.121 address is specified in the option. By default Gcom_hpad listens for incoming calls from any remote DTE. Note that if this option is used and if the incoming call X.25 packet does not contain the "calling address" then the call will never be accepted by this instance of Gcom_hpad.
–Q Log-Size	Sets the size, in bytes, of the log file created by Gcom_hpad. If this parameter is left unspecified the log will grow indefinitely. If this parameter is specified the log will be maintained in circular fashion.
–h	Prints out an abbreviated version of the parameter syntax.
–s	The rest of the command line beyond this parameter is used as the command to start on the slave side of the pseudo-tty associated with each X.25 call. In the absence of this parameter the default used is "/bin/login -h X.25PAD".

Operation

The `Gcom_hpad` listens for incoming X.25 calls using the listen pattern provided by the user, or the default pattern of "*". It forks a child process to handle each new X.25 call.

Once the call has been accepted it sends an X.29 "set parameters" Q-bit packet to the remote PAD setting the X.3 parameters to the desired settings. The default settings for the X.3 parameters are as follows. Parameter references not shown are not included in the Q-bit packet sent to the remote PAD.

X.3 Reference	Value	Description
2	0	No local echo
3	0xFF	Forward on every character
4	0	Disable idle timer
5	2	Control flow of data from terminal
12	0	No control of flow of data to terminal by the PAD.
13	0	No linefeed insertion
15	0	No editing by the PAD
16	0	No character delete character
17	0	No line delete character
18	0	No line display character
20	0xFF	No character echo
22	0	Page wait disabled

These settings can be modified via the "`-p`" command line argument.

The child process exits whenever it receives a "hangup" on the pseudo-tty, usually indicating that the process on the slave side of the pseudo-tty has exited. It also exits when the X.25 virtual circuit is cleared.

Gcom_tpad

The `Gcom_tpad` (terminal PAD) program contains an implementation of the CCITT X.3, X.28 and X.29 protocols. It is an interactive program that the user runs from a shell prompt. The stdin and stdout files to the user's terminal play the role of the asynchronous port in the CCITT model PAD. There are no modem signal manipulations performed on the user's terminal port.

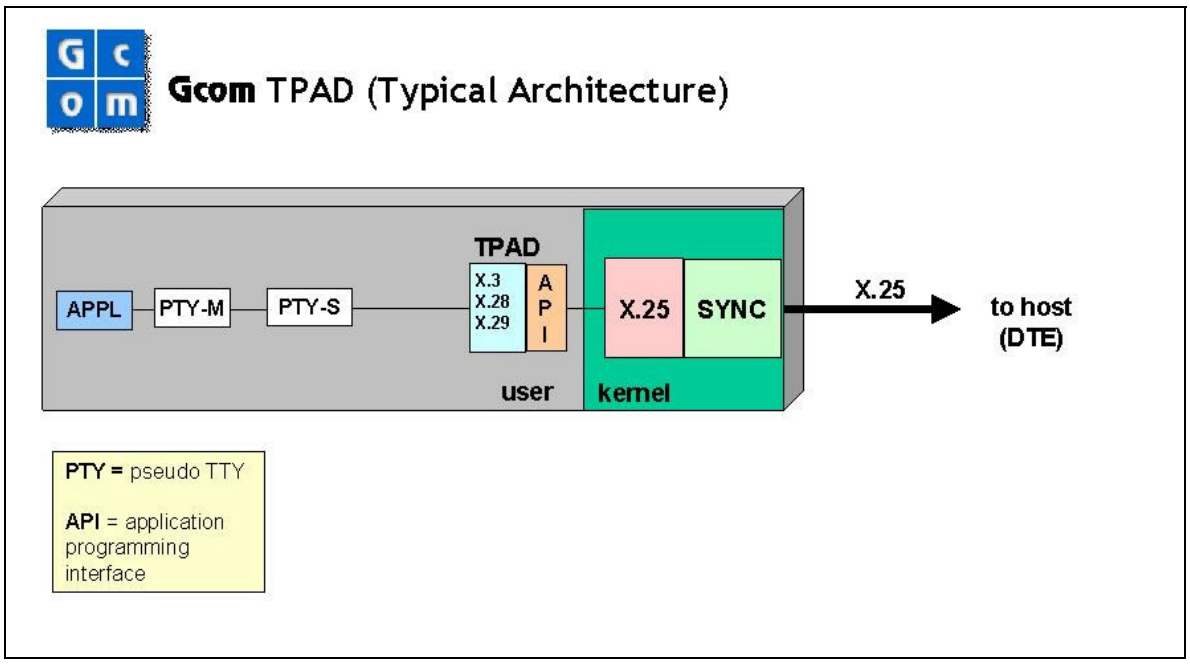
The user interface used by the X.28 portion of `Gcom_tpad` is a variant on that which is defined by the standard. The style of this interface is something of a mixture of X.28 and the old GTE Telenet PAD user interface.

`Gcom_tpad` uses the GCOM NPI API interface to communicate with GCOM'S X.25 protocol, which resides in a kernel mode driver. Thus, the PAD functions run in user space and the underlying X.25 protocol runs in the kernel. This means, among other things, that virtual circuits managed by GCOM'S X.25 protocol can be used for other purposes simultaneously with the circuits used by **`Gcom_tpad`**.

The `Gcom_tpad` program manages exactly one session with the user and exactly one X.25 virtual circuit to the remote DTE (host). `Gcom_tpad` initiates X.25 calls to the remote host. It does not listen for or accept incoming X.25 calls.

Any number of copies of the `Gcom_tpad` program can be run to manage any number of connections to remote X.25 hosts, subject to the limitations of X.25 virtual circuits.

The `Gcom_tpad` and `Gcom_hpad` can both be operating at the same time and will share the virtual circuits of the X.25 access lines.



Invocation

The `Gcom_tpad` is invoked using the following command line syntax:

`Gcom_tpad` *options*

The values for *options* are:

–dDebug-Mask	Set the internal debug mask to the given value. These values are internal and should be used at the direction of GCOM support personnel.
–fConn-Flags	Set the flags to be used in the NPI connect request primitive. The value of this parameter consists of the logical *or* of individual bit values. The value **0x0001** is used to request that the D-bit service of X.25 be allowed for the X.25 call.
–FBind-Flags	Set the flags to be used in the NPI bind request primitive. The value of this parameter consists of the logical *or* of individual bit values. While a number of these options exist, none of them are useful to `Gcom_tpad`. This parameter is mainly for future expansion.
–LLog-Name	Sets the name of the log file created by `Gcom_tpad`. The default name is `Gcom_tpad.log`.

***Note*:** When running multiple copies of `Gcom_tpad` you should name the log files differently for the different copies of the program, or run the program from different directories so as to avoid using the same log file.

–OLog-Options	NPI log options. These are bit-encoded. They are described in the *NPI API Guide* and in the file <gcom/npiapi.h>. The default value is **0x0013**.
–QLog-Size	Sets the size, in bytes, of the log file created by `Gcom_tpad`. If this parameter is left unspecified the log will grow indefinitely. If this parameter is specified the log will be maintained in circular fashion.
–h	Prints out an abbreviated version of the parameter syntax.

Operation

The `Gcom_tpad` program reads characters in raw mode from the user's terminal via stdin and feeds them into GCOM'S X.3/X.28/X.29 PAD module as if they were keystrokes from a directly connected asynchronous terminal. The PAD module assembles the characters into packets and forwards them to the remote host via X.25.

In the reverse direction, `Gcom_tpad` reads X.25 packets from the GCOM NPI kernel driver, which contains the X.25 protocol code, and forwards them to the GCOM PAD module. The PAD module processes the packets in accordance with the CCITT standards and then outputs them to the user's terminal via stdout.

The PAD module is contained within the `Gcom_tpad` program, which is an interactive user-space process. The X.25 module is contained within the GCOM NPI kernel mode driver. `Gcom_tpad` uses GCOM'S NPI API to interface to the X.25 module in the kernel.

When `Gcom_tpad` is first run it is in interactive command mode with the user. The user uses the CALL command to place an X.25 call to a remote host. Once the call has been accepted the PAD enters data mode and shuttles characters back and forth between the user's terminal and the remote X.25 host, as described above.

`Gcom_tpad` reacts to an "invitation to clear" X.29 message from the remote X.25 host by clearing the X.25 call to the remote host. If `Gcom_hpad` is operating on the remote host, such a message will be sent out when the user logs out. Thus, it is usually not necessary to clear the X.25 call by "recalling" the PAD and issuing a CLR command.

The `Gcom_tpad` program maintains a log file whose default name is `Gcom_tpad.log`. It logs all calling, clearing and reset NPI messages in this log file. In addition there are command line options that will cause `Gcom_tpad` to write voluminous debugging information into its log file, and options to control the size of that file. These options are most useful for troubleshooting purposes.

Usually there are no command line options required to run `Gcom_tpad`. Under highly unusual circumstances one may find the bind and connect options to be useful.

`Gcom_tpad` implements flow control coupling between the user's terminal and X.25 (via NPI). It uses the *poll()* system call to ensure writability on the user's terminal and to

X.25 and asserts appropriate back-pressure on the PAD code. This allows for high volume and high data rate transfers to take place through the `Gcom_tpad` program.

When using `Gcom_tpad` to connect to a host running `Gcom_hpad`, the `Gcom_hpad` will cause `Gcom_tpad` to cease echoing characters and to forward all keystrokes to the host as they are read from the user's terminal. This allows the Unix host computer to control the echoing of keystrokes on the user's terminal.

> ***Note*:** Even though echoing and processing of characters is suppressed in the PAD module, the PAD module still recognizes the "PAD recall character" (default "@") as an escape back to command mode in the PAD. Thus, whenever the user types, or otherwise inputs, the "@" character, the PAD module will enter command mode for one line of input.

If you wish to defeat this mechanism, you need to set X.3 parameter #1 to the value of zero. The PAD command to accomplish this is as follows:

```
set 1:0
```

Once this command has been entered, you can no longer escape back to command mode in the PAD when your terminal is connected to a remote host via an X.25 virtual circuit. If you have the need to accomplish bulk data transfers via the PAD, towards the remote host, you can send the data more efficiently by using the following PAD command, after calling the remote X.25 host.

```
set 1:0,3:0,4:1
```

In order to issue this command, you must first call the remote host and log in. Then use the PAD recall character ("@") to escape to command mode. Then type the above command. This will put your terminal in a mode in which you can no longer escape to command mode and in which the PAD gathers input characters from your terminal until a packet fills up or until 50ms of idle time passes before sending the packet to X.25. This maximizes the efficiency of the X.25 line for sending data from your terminal to the remote host. You can operate your PAD session in this mode all the time, but you will find a certain "looseness" in the echoing of characters due to the 50ms delay in forwarding individual keystrokes.

PAD Commands

Command Usage

This section describes the syntax and semantics of the commands that are used to communicate with the Rsystem® PAD system.

Notation Guide

Symbols:

xxx	Indicates that the italic typeface string is a meta-symbol describing a category of symbols, any one of which may substitute for xxx.
[*xxx*]	Indicates that the string enclosed in brackets is optional in only that location in the command line.
{ *xxx* }	The same as [xxx], but may appear anywhere on the command line, not just in the location in the example.
x | *y*	Indicates that x or y may be used
xxx	**Boldface** type indicates that the string **xxx** must be entered as indicated.

In the following summaries, the shortest unique abbreviation of each command is underlined.

Categories:

address	Consists of 1 to 15 decimal digits
facilities	Consists of zero to 63 hexadecimal digits
user-data	Consists of zero to 12 characters
refn	A valid X.3 parameter reference (in decimal)
valn	A valid X.3 parameter value (in decimal)
tab-num	A decimal number from zero to 16 inclusive
exp-num	Either zero or 1
profile	A valid profile identifier
chn	A character -- currently must be entered in decimal

Commands may be entered in upper of lower case, or a mixture of the two. In the following summaries, the shortest valid abbreviation of the command is underlined. The commands are shown in UPPER CASE, but they are recognized when typed in lower case as well.

Local PAD Commands

The commands in the following list operate on the state of the local PAD. *See Remote Commands* for commands that operate on the remote X.29 host.

CALL [?] | [*address* } [***P** | ***D***user-data*]]

Generates a call request packet. This is the equivalent of the CCITT "selection" command. *address* is the called address that is inserted in the call request packet. It will be inserted in call request packets generated by **CALL** commands without an *address* argument until another **CALL** command with an *address* argument is issued.

Any text following the ***D** or ***P** will be inserted into the last twelve bytes of the user data field. If *D is used characters typed will be echoed as they are entered. If ***P** is used, they will not be echoed as they are entered (useful is the user data field is to carry a password.)

If the **CALL** command is entered with only a question mark, the PAD will display the address the PAD would use if the **CALL** command were entered with no address.

For example:

CALL 22200064 *DME
C 22200064

CLR

Generates a clear request packet.

For example:

CL

FACILITIES [* | *facilities*]

Stores facilities to use in subsequent **CALL** commands. The *facilities* entered on the command line are input-converted from hexadecimal and become the byte sequence inserted in the facilities field of outgoing call request packets until another facilities command is issued.

The **FACILITIES** command, with no argument, displays the current facilities.

The command **FACILITIES *** clears the current facilities so the default will be used. The default facilities are window size 2 and packet size 128 (420202430707).

> ***Note***: The *facilities* command simply stores all bytes from the first syntactically significant character after the command identifier to the end of the line. It makes no attempt to ensure that the facilities argument consists of legitimate facilities values, or even that it is a number.

For example:

FACILITIES 420505	(window size = 5)
F 430808	(packet size = 256)
FACIL 0101	(reverse charges)
FACILITIES0101420505430808	(all of the above)
F *	(use default facilities)

FULL

Selects full duplex mode (see **HALF**, below.) If the PAD session is already in full duplex mode, **FULL** has no effect.

HALF [*] | [[-] *ch1,ch2,...,chn*]

Selects half-duplex mode, and specifies characters which are to be echoed. In half duplex mode, most characters are not echoed. If linefeed insertion is enabled, the inserted linefeeds will be sent to the terminal. If tab expansion is enabled, the resulting spaces will be sent to the async device. If linefeed or carriage return padding is enabled, the padding will be sent to the terminal, even if the padded character is not echoed.

Also, the HALF command allows the user to select characters that should be echoed -- a type of reverse echo mask. For example, HALF 13,10 selects carriage return and linefeed to be echoed. HALF * deselects all characters, so that none will be echoed. A character list proceeded by a hyphen "-" (e.g., HALF - 10,13) deselects only the characters in the list. The HALF command with no arguments sets half duplex mode without altering the characters which have been selected for echo via previously entered HALF commands.

For example:

HALF	(select half-duplex mode)
HA 13,10,9	(select half duplex mode, echo carriage return, linefeed, and tab)
HA - 9	(select half duplex mode, disable previously enabled echo of tab)
HALF *	(select half duplex mode, disable previously enabled echo for all characters)

HELP

Displays the list of PAD commands.

For example:

H

INTERRUPT

Generates an interrupt packet.

For example:

I

LISTEN [**ADDR**=*address* | **DATA**=*user-data*]

Specifies the match pattern for accepting an incoming call. The *address* argument is a (sub)address to match to determine if the incoming call is intended for this async port. The *user-data* argument is a pattern to match against the last 12 bytes of the user data field of incoming calls to determine if the incoming call is intended for this async port.

For example:

LISTEN ADDR=22
L A=22

Note: This command is not implemented in this version of the PAD.

PAR? [*ref1* [*,ref2,...,refn*]]

Displays the current value of one or more X.3 parameters.

For example:

PAR (display all X.3 parameters)
PA 13,16,17 (display values of X.3 parameters 13, 16, and 17)

PROF [*profile* | ?]

Configures the PAD using the values associated with a PAD parameter profile. **PROF** by itself displays the currently active profile. **PROF ?** - displays the currently active profile followed by lists of available profiles. **PROF *profile*** - activates that profile (i.e., sets the PAD's X.3 and extra parameters to the values associated with the named profile.)

For example:

PR CRT (enables CRT profile)

RESET

Generates a reset request packet with zero cause (DTE originated) and zero diagnostic (no additional information.)

For example:

R

SET [*ref1:val1* [*,ref2:val2,...,refn:valn*]]

Sets one or more X.3 parameters to specified values.

For example:

SE 13:5,16:8 (enables linefeed insertion on echo and output and make backspace the character delete character)

SET? [*ref1:val1* [*,ref2:val2,...,refn:valn*]]

Sets one or more X.3 parameters to specified values and then displays those parameters and their values.

For example:

SET? 13:4 (enable linefeed insertion on echo, then display the newly set value of parameter 13)

STATUS

Interrogates virtual call status.

For example:

S

TABS { **LCL** *tab-num* } { **REM** *tab-num* } { **EXP** *exp-num* }

Sets and reads three nonstandard parameters that control tab expansion. These parameters are not accessible by the remote host via Q-bit packet PAD commands. The arguments have the following meanings:

EXP Enables (1) or disables (0) expansion of tabs locally by the PAD to the number of blanks specified by the **LCL** argument.

LCL Sets the number of columns to which tabs are expanded locally, i.e., to the terminal on echo and output from the network. If the **EXP** parameter is zero, and the **LCL** parameter is nonzero, then LCL means the number of columns to which the terminal itself is expanding tabs it receives. This situation is useful for tracking absolute line position for line folding and destructive line deletion. Zero means no expansion.

REM Sets the number of columns to which tabs are expanded remotely i.e., on input from the terminal towards the network. Zero means no expansion.

For example:

TABS LCL 8 EXP 1	(expand tabs to terminal to eight blanks)
T L 4 REM 4 E 1	(expand tabs to both terminal and remote to four blanks)

Remote Commands

The remote commands generate Q-bit data packets containing PAD messages as defined in CCITT recommendation X.29. These commands are useful when the remote DTE is not a host with X.25 capability, but some piece of equipment hooked to a PAD. The remote commands can be used to interrogate and change the configuration of the remote PAD, to clear a call without loss of data, and to test the X.29 functionality of the remote PAD.

RPAR? [*ref1* [*,ref2,...,refn*]]

Syntactically identical and functionally similar to the PAR? command explained above. Displays the current value of one or more X.3 parameters on a remote PAD. It interrogates the remote PAD by sending it a Read PAD message and displaying the contents of the resulting Parameter Indication PAD message.

For example:

RPAR?	(display all of the remote PAD's X.3 parameters)
RPA 13,16,17	(display values of the remote PAD's X.3 parameters 13,16, and 17)

RPROF [*profile* | **?**]

Syntactically identical and functionally similar to the **PROF** command, explained above. It configures the remote PAD using the values associated with a PAD parameters profile defined on the local PAD. **RPROF** by itself has no effect. **RPROF ?** displays the local PAD's currently active profile followed by lists of locally available profiles. **RPROF profile** activates that profile on the remote PAD (i.e., sets the remote PAD's X.3

parameters to the values associated with the named profile). **RPROF** sets the remote PAD's parameters by sending it a Set and Read PAD message.

For example:

RPR CRT	(enable CRT profile on the remote PAD)

RSET [*ref1:val1* [,*ref2:val2,...,refn:valn*]]

Syntactically identical and functionally similar to the SET command described above. Sets one or more of the remote PAD's X.3 parameters to specified values by sending a Set PAD message.

For example:

RSE 13:5,16:8	(enable linefeed insertion on echo and output; make backspace the delete character on the remote PAD)

RSET? [*ref1:val1* [,*ref2:val2,...,refn:valn*]]

Syntactically identical and functionally similar to the **SET?** command described above. Sets one or more of the remote PAD's X.3 parameters to specified values and then displays those parameters and their values.

For example:

RSET? 13:4	(enable linefeed insertion on echo on the remote PAD, then display the remote PAD's newly set value of parameter 13).

RICLR

Sends an Invitation to Clear PAD message to the remote PAD, which should cause it to clear the virtual call after all data that it previously received has drained into the remote terminal.

For example:

RI

Parameter Miscellany

Data Forwarding Characters (ref #3)

The GCOM PAD implementation of parameters 3 is somewhat nonstandard. The standard makes it impossible to specify data forwarding on every character by omitting the punctuation characters from the data forwarding ranges. However, the GCOM PAD accepts the hex 80 bit (decimal 128) and gives it the meaning "all characters not covered by the other ranges." Thus a parameter 3 value of 255 means "data forwarding on all characters."

Echo Mask (ref #20)

The GCOM PAD implementation of parameter 20 is somewhat nonstandard. The hex 40 bit (decimal 64) is accepted, but has no meaning. Echo mask of editing characters does not function.

Parity Treatment (ref #21)

This parameter has no effect on `Gcom_tpad`. It does not cause a change in the settings of the terminal attributes of the controlling terminal.

Page Wait (ref #22)

The GCOM PAD implementation of parameter 22 is nonstandard. The page wait linefeed count is reset by only two events:

- Echo of a linefeed from the terminal
- Cancellation of the page wait condition

The page wait condition is canceled by receipt of any character from the terminal. The character that cancels page wait is discarded, and data flow is restarted towards the terminal. Thus, the linefeed count tracks the number of linefeeds in service signals and output since the last echoed linefeed. If you wish to type ahead without scrolling the screen that is frozen because of page wait, you can enter an XOFF (Ctrl S) and proceed.

Half Duplex v.s. Echo

At first glance, it may appear that selecting half-duplex mode is the same as disabling echo. However, the difference is great; when echo is disabled, it radically decreases the amount of processing the PAD must do on every character, and certain interface functions (line folding, tab expansion, linefeed insertion, CR and LF padding, and destructive character and line deletion) cannot be performed. This is the most efficient mode for file transfers.

In half-duplex mode with echo enabled, the PAD does most of the work of echoing and then discards the data instead of sending it to the terminal. This arrangement makes it possible to provide line folding, tab expansion linefeed insertion, CR and LF padding, and destructive character and line deletion while in half-duplex mode.

Terminals Which Expand Tabs

The PAD's view of the current cursor position on your screen is kept accurate by setting the tab expansion parameter (**EXP**) to zero and the local tab expansion parameter (**LCL**) to the number of columns to which your terminal expands tabs. This setting allows the PAD to provide correct line folding, destructive line deletion, and destructive character deletion.

Flow Control

The GCOM Rsystem PAD has a variety of flow-control mechanisms that are discussed more thoroughly in the *Rsystem PAD Module Manual*.

In brief, the main points of flow are:

1. If the PAD is unable to flow-control stop the terminal, the PAD may start buffering characters without processing them. These characters are not discarded, but their processing is delayed until the flow control blockage is relieved. When this buffering occurs, an interactive user may notice that his/her keystrokes are no longer being echoed. If the flow control blockage does not clear and the PAD continues to receive characters from the user, the PAD may reach the point at which it must discard data for lack of buffer space. Any data characters received will then be ignored, and the PAD will send a character to the terminal to notify the user that data is being discarded. The character sent will usually cause the bell on the terminal to ring.

2. If the PAD reaches the point at which it discards data, it may force the flow of echo and output to the async device in order to free buffers for use if the user continues to send it commands that tie up buffers. Under such conditions it becomes impossible to flow-control stop the PAD, and input and output may end up on the same line.

PAD Command Summary

CALL [?] | [*address* } [***P** | ***D***user-data*]]
CLR
FACILITIES [* | *facilities*]
FULL
HALF [*] | [[-] *ch1,ch2,...,chn*]
HELP
INTERRUPT
LISTEN [**ADDR**=*address* | **DATA**=*user-data*]
PAR? [*ref1* [*,ref2,...,refn*]]
PROF [*profile* | ?]
Reset
SET [*ref1:val1* [*,ref2:val2,...,refn:valn*]]
SET? [*ref1:val1* [*,ref2:val2,...,refn:valn*]]
STATUS
TABS { **LCL** *tab-num* } { **REM** *tab-num* } { **EXP** *exp-num* }

X.3 Parameters

1 **PAD Recall Character**

0	---None
1	---DLE
32-126	---ASCII character

2 **Echo**

0	---Disable
1	---Enable

3 **Data forwarding character**

7	6	5	4	3	2	1	0

Bit	Function
0	All alphanumerics
1	CR
2	ESC, BEL, ENQ, ACK
3	DEL, CAN, DC2
4	ETX, EOT
5	HT, LF, VT, FF
6	All other ASCII control
7	All other characters

0 = None

4 **Idle Timer Delay**

0	---Disable timer
1-255	---n/20 seconds

5 **Ancillary device controll**

0	---No X-ON/X-OFF
1	---On data transfer only
2	---On data transfer and command

6 **Service Signals**

7	6	5	4	3	2	1	0

Bit	Function
0	Service signals other *than prompt PAD service* sent in standard format
1	Not used
2	Prompt PAD service sent in standard format
3	Network dependent service signals
4	Not used
5	Not used
6	Not used
7	Not used

0 = None

7 **Action on *break***

7	6	5	4	3	2	1	0

Bit	Function
0	*Interrupt* packet
1	Reset
2	*Indication of break* PAD message
3	Escape from *data transfer* state
4	Discard output to terminal
5	Not used
6	Not used
7	Not used

0 = No action

8 **Discard output to terminal**

0	---Do not discard
1	---Discard

9 **Padding after CR**

0-255	---Number of padding characters appended when CR sent to terminal.

10 **Line Folding**

0	---No line folding
1-255	---Number of graphic characters sent by PAD before line folding.

11 **Binary Speed (read only)**

0	---110
1	---134.5
2	---300
3	---1200
4	---600
5	---75
6	---150
7	---1800
8	---200
9	---100
10	---50
11	---75/1200
12	---2400

13	---4800
14	---9600
15	---19,200
16	---48,000
17	---56,000
18	---64,000

12 **Flow control of PAD by terminal (X-ON/XOFF)**

0	---Disable
1	---Enable

13 **Linefeed Insertion**

7	6	5	4	3	2	1	0

Bit	Function
0	After CRS in data stream to terminal
1	After CRS from terminal
2	After CRS in echo terminal
3	Not used
4	Not used
5	Not used
6	Not used
7	Not used

0 = None

14 **Linefeed padding** 0-255 ---Number of padding characters sent to terminal after LF in data-transfer state.

15 **Editing**

0	---Disable
1	---Enable

16 **Character delete** 0-127 ---Any ASCII character.

17 **Line delete** 0-127 ---Any ASCII character.

18 **Line display** 0-127 ---Any ASCII character.

19 **Editing PAD service signals**

0	---Disable
1	---Printing terminal
2	---Display terminal
8	---Use ASCII BS
32-126	---Use ASCII graphics character

Bit	Function

20 **Echo mask**

7	6	5	4	3	2	1	0

0 CR
1 LF
2 VT, HT, FF
3 BEL, BS
4 ESC, ENQ
5 ACK, NAK, STX, SOH, EOT, ETB, ETX
6 Editing characters
7 All other ASCII control characters
Characters not echoed.

21 **Parity treatment**

0	---Ignore parity
1	---Generate only
2	---Check and generate

22 **Page wait**

0	---Disable
1- 255	---Number of LFs before page wait

Clear Cause Codes

Code (hex)	Cause of Clear
01	Number busy
03	Invalid Facility Request
05	Network Congestion
09	Out of Order
0B	Access Barred
0D	Not Obtainable
11	Remote Procedure error
13	Local Procedure Error
15	RPOA Out of Order
19	Reverse Charging Acceptance Not Subscribed
21	Incompatible Destination
29	Fast Select Acceptance Not Subscribed
39	Ship Absent
C1	Gateway-detected Procedure Error
C3	Gateway Congestion

Reset Cause Codes

Code (hex)	Cause of Reset
01	Out of Order
03	Remote Procedure Error
05	Local Procedure Error
07	Network Congestion
09	Remote DTE Operational

0F	Network Operational
11	Incompatible Destination
1D	Network Out of Order
C1	Gateway-detected Procedure Error
C3	Gateway Congestion
C7	Gateway Operational

Selected X.25 Diagnostic Field Values

hex	decimal	Diagnostic
0	0	No additional information
1	1	Invalid P(S)
2	2	Invalid P(R)
10	16	Packet type invalid
11	17	For state r1
12	18	For state r2
13	19	For state r3
14	20	For state p1
15	21	For state p2
16	22	For state p3
17	23	For state p4
18	24	For state p5
19	25	For state p6
1A	26	For state p7
1B	27	For state d1
1C	28	For state d2
1D	29	For state d3
20	32	Packet not allowed
21	33	Unidentifiable packet
22	34	Call on one-way LC
23	35	Invalid packet type on a PVC
25	37	REJECT not subscribed to
26	38	Packet too short
27	39	Packet too long
29	41	Restart packet with nonzero LC
2B	43	Unauthorized interrupt confirmation
2C	44	Unauthorized interrupt
2D	45	Unauthorized reject
30	48	Timer Expired
31	49	For INCOMING CALL (or for DTE timer expired)
32	50	For CALL REQUEST
33	51	For CLEAR INDICATION (or for DTE timer expired or retransmission count surpassed for CLEAR REQUEST
34	52	For RESET INDICATION (or for DTE timer expired or retransmission count surpassed for RESET REQUEST)
40	64	Call setup, call clearing, or registration problem
41	65	Facility/registration code not allowed

42	66	Facility parameter not allowed
43	67	Invalid called address
44	68	Invalid calling address
45	69	Invalid facility/registration length
46	70	Incoming call barred
47	72	Call collision
48	73	Duplicate facility requested
49	74	Nonzero address length
4A	74	Nonzero address length
4B	75	Nonzero facility length
4C	76	Facility not provided when expected

ASCII Chart

Decimal	Character	Decimal	Character
0	NUL	64	@
1	SOH	65	A
2	STX	66	B
3	ETX	67	C
4	EOT	68	D
5	ENQ	69	E
6	ACK	70	F
7	BEL	71	G
8	BS	72	H
9	HT	73	I
10	LF	74	J
11	VT	75	K
12	FF	76	L
13	CR	77	M
14	SO	78	N
15	SI	79	O
16	DLE	80	P
17	DCI	81	O
18	DC2	82	R
19	DC3	83	S
20	DC4	84	T
21	NAK	85	U
22	SYN	86	V
23	ETB	87	W
24	CAN	88	X
25	EM	89	Y
26	SUB	90	Z
27	ESC	91	[
28	FS	92	\
29	GS	93	]

30	RS	94	^
31	US	95	_
32	SPACE	96	`
33	!	97	a
34	"	98	b
35	#	99	c
36	$	100	d
37	%	101	e
38	&	102	f
39	'	103	g
40	(	104	h
41	)	105	i
42	*	106	j
43	+	107	k
44	,	108	l
45	-	109	m
46	.	110	n
47	/	111	o
48	0	112	p
49	1	113	q
50	2	114	r
51	3	115	s
52	4	116	t
53	5	117	u
54	6	118	v
55	7	119	w
56	8	120	x
57	9	121	y
58	:	122	z
59	;	123	{
60	<	124	|
61	=	125	}
62	>	126	~
63	?	127	DEL