term

A terminal program for Amiga computers

ARexx-interface and ARexx-commands explained
7 February 1995

by Olaf Barthel

The program ‘term’ and the data received/sent by it must not be used for the following purposes:

1. The construction, development, production or testing of weapons or weapon systems of any kind.
2. The construction, development, production or use of plants/installations which include the processing of radioactive/fissionable material.
3. The training of persons to deal with the abovesaid actions.

Listen to your conscience.

1 Changes

Previous 'term' releases would use a different ARexx host interface implementation. In order to conform to Commodore-endorsed user interface style guidelines it was redesigned from scratch for version 3.0. The design and implementation of the ARexx host interface was suggested by the *Amiga User Interface Style Guide* and strongly influenced by Martin Taillefer's *TurboText* ARexx host interface.

Not a simple command has 'survived' the revision, the new implementation is no longer compatible with its predecessors, so existing ARexx scripts will have to be adapted or even entirely rewritten.

'term' no longer distinguishes explicitely between asynchronous and synchronous commands (i.e. commands which force the main program to wait and commands which need not bother the main program as the ARexx handler process is able to execute them). As of this writing it is safe to assume that almost any command will be processed by the main program, exceptions are noted.

2 term and ARexx

This document describes the ARexx(tm)[1] commands supported by 'term'. This is not intended to be an introduction to the language itself. Rexx was developed by Mike F. Cowlishaw on an IBM/SP system and ported to the Amiga by William S. Hawes.

ARexx (or Amiga Rexx) is a commercial product which is included with the AmigaDOS 2.0 Enhancer Package. If you need a good introduction and description of the language, try to get a hold of the book *The REXX Language A Practical Approach to Programming* by M.F. Cowlishaw, available from Prentice-Hall International, Inc.

The section entitled Section 2.1 [Command execution], page 3 gives a brief introduction how to write and run ARexx commands. For more information refer to the Release 2 Users Manual *Using the System Software*.

By default 'term' opens an ARexx host by the name of `TERM` (accessable via `address term`). If more than a single 'term' process is running on your machine, the name of the host will be adapted to the number of the program (i.e. the first programm will use `TERM`, the second one will use `TERM.1`, the third one `TERM.2`, etc.). The default name can be overridden by invoking the program with certain parameters (see main program documentation). The name of the host is displayed in the status window (see main program documentation).

2.1 Command execution

In order to invoke any command supported by 'term' one usually has to address the host explicitely:

```
/* Address the 'term' host. */

ADDRESS term

/* Invoke the 'beepscreen' commmand. */

BEEPSCREEN
```

However, if an ARexx script is invoked directly by the 'term' main program, the script will by default address the main program it was invoked by.

[1] ARexx is a registered trademark of Wishful Thinking Development Corp.

Most commands will return results or error codes on failure. To enable result codes, one has to use the `options results` command. The results returned by commands will be placed in the `result` variable:

```
/* We assume that the script will address the host it was invoked from.
 *
 * Enable command results.
 */

OPTIONS RESULTS

/* Request a string from the user. */

REQUESTSTRING DEFAULT 'anything' PROMPT 'Enter anything'

/* Did the user cancel the requester? */

IF rc ~= 0 THEN
   SAY 'user cancelled requester'
ELSE
   SAY result /* Output the result . */
```

Failure codes will always be returned in the `rc` variable (see previous example).

In case of failure (variable `rc` >= 10), 'term' will leave an error code in the `term.lasterror` variable:

```
/* Enable command results. */

OPTIONS RESULTS

/* Produce an error by not supplying any arguments. */

STOPBITS

/* Display the error code. */

SAY term.lasterror
```

Rexx tries to tokenize any command parameters, this process involves promoting them to all upper case letters and checking for illegal characters. This feature inhibits the use of the **:** (colon) and blank space characters in parameter names unless the corresponding arguments are enclosed in quotes. To make things even more complicated, the parser will not always accept parameters to contain blank spaces. If a command template accepts the entire command line (such as `TEXT/K/F`) a parameter can include any number of blank spaces. A command template to accept just a single parameter (such as `TEXT/K`) requires double quotes if blank spaces are included. Text such as `tea or coffee?` thus becomes `'"tea or coffee?"'`.

```
/* The following command will fail to send the file `ram:foobar' as the colon
 * in the path name will cause an error:
 */

SENDFILE ram:foobar

/* Here is how to do it correctly: */

SENDFILE 'ram:foobar'

/* The following command will fail to send the file `foo bar' as the
 * file name is treated as two single files:
 */

SENDFILE foo bar

/* The next line will still fail to send the file `foo bar'
 * as the ARexx parser will split the argument into two
 * parameters.
 */

SENDFILE 'foo bar'

/* Here is how to do it correctly: */

SENDFILE '"foo bar"'

/* The following command will not transmit the string `Hello sailor'
 * across the serial line as the single words will be capitalized,
 * they will be transmitted as `HELLO SAILOR':
 */

SEND Hello sailor

/* Here is how to do it correctly: */

SEND 'Hello sailor'
```

3 Stopping a command

Programs and commands sometimes fail to do what the user is expecting them to do which makes it necessary to bring program/command execution to a stop. A common ARexx script to call no external functions or host commands one can be halted in the following ways:

1. Executing the `HI` command (located in the '`SYS:rexxc`' drawer) from Shell. This command will attempt stop *all* currently running ARexx scripts.
2. If the ARexx script to be executed runs in an environment to sport an output window, activate the window and press the `Control` + `C` keys. A break signal will be sent to the ARexx script, causing it to stop as soon as possible.

With host environments such as 'term' it may not always be possible to abort a command using the simple measures described above. As for 'term' any command to wait (such as the Section 4.33 [READ], page 33, Section 4.13 [DELAY], page 19 or Section 4.59 [WAIT], page 51 commands) can be aborted by sending 'term' itself a break signal in the following fashion:

1. If the 'term' program is still attached to a Shell output window, activate the window and press the `Control` + `D` keys.
2. If the 'term' program was invoked from a Shell but is no longer attached to it, enter `status command term` from Shell, remember the number printed, then enter `break <number>` with `<number>` being the number returned by the '`status`' command.
3. Press the hotkey combination configured in the program hotkey settings (see main program documentation). The default is `Right Shift` + `Left Shift` + `Escape`. This will cause a break signal to be sent to the 'term' program.

4 Commands

The commands supported by 'term' are listed in a table of the following form:

`Format:`

The command name with its possible calling parameters. In this table parameters are enclosed in brackets and braces, separated by commas and vertical bars; *do not type these special characters along with the parameters!*:

`< > (Angle brackets)`
: Angle brackets enclose parameters whose contents **must not** be omitted in order to make the command work properly.

`[ ] (Square brackets)`
: Square brackets enclose optional parameters.

`{ } (Curly braces)`
: Curly braces enclose items which can be repeated a number of times, such as file name lists.

`| (Vertical bar)`
: Vertical bars separate alternative, mutually exclusive options.

`, (Comma)`
: Commas separate multiple applicable options.

`Template:`

The command template, similar to the command templates employed by AmigaDOS Shell commands. Possible templates are:

`<Parameter>/A`
: The parameter must **always** be included in order to get accepted.

`<Option>/K`
: The option's **keyword** must be given.

`<Option>/S`
: This option works as a **switch**. If this option keyword is included the switch is on, else it is off.

`<Option>/N`
: A **numeric** parameter is expected.

`<Option>/M`
: **Multiple** parameters are accepted.

`<Text>/F`
: The text must be the **final** parameter on the command line.

`, (Comma)`
Indicates that the command takes no parameters.

`Purpose:`
Briefly describes what the command will do.

`Specifications:`
Describes the command and its possible uses in more detail.

`Result:`
The type of the command result code if any.

`Warning:`
If the command can return with a warning and when.

`Example:`
An example code fragment to illustrate how to use the command. Commands and keywords are given in upper case, the names of variables and command arguments are given in lower case. Where a single command line would not fit into a single line on the screen, an ellipsis ('...') is meant to join the broken line.

4.1 The ACTIVATE command

`Format:`
ACTIVATE

`Template:`
,

`Purpose:`
De-iconifies the program, brings the main window to the front and makes it active.

`Specifications:`
The program can be put to sleep using the Section 4.12 [DEACTIVATE], page 18 command, to bring it back to proper operation, use the `ACTIVATE` command. If this command is invoked while the program is not asleep, it will cause the main window to be brought to the front and activated.

`Result:`
-

`Warning:`
-

`Example:`
```
/* This is how the main programm can be (re-)activated: */

ACTIVATE
```

4.2 The ADDITEM command

`Format:`

ADDITEM [To] <Upload|Download|Dial|Wait> [Before|After] [Command <Command for trap list>] [Response <Response text>] [Phone <Entry number, name or wildcard pattern>] [Name <Name>]

`Template:`

TO/A,BEFORE/S,AFTER/S,RESPONSE/K,COMMAND/K,PHONE/K/F,NAME/K/F

`Purpose:`

Inserts an item (a name, a phone number, text, etc.) before or after the currently selected list item.

`Specifications:`

'term' maintains a number of lists, these are:

`Upload list`

The list of files to be uploaded.

`Download list`

The list of files the program has downloaded.

`Dial list`

The list of phone numbers or phone book entries to be dialed.

`Wait list`

The list of texts the Section 4.59 [WAIT], page 51 command is to wait for.

New items can be added to the list with the `ADDITEM` command. The upload list expects the names of files the Section 4.52 [SENDFILE], page 47 command is to transfer. It makes little sense to add files names to the download list as the 'term' main program maintains it and adds the names of files received to it, but it is still possible. The wait list expects text lines the Section 4.59 [WAIT], page 51 command will look for in the terminal input stream. A wait list entry added using the `RESPONSE` keyword will if found in the input data stream cause the response text to be immediately sent to the remote. *Note: a wait list entry to make use of the* `RESPONSE` *keyword will be handled by the Section 4.59 [WAIT], page 51 command, the ARexx script will not notice if this list entry was found or not.*

The dial list accepts a number of different parameters:

`Phonebook entry numbers`

These are passed using the `Phone` parameter which should be a numeric value as it is used as an index to pick the corresponding entry from the phone book.

`Phonebook entry names`

These are also passed using the `Phone` parameter which can be a proper name or a wildcard pattern.

`Phone numbers`
These are passed using the `Name` parameter.

List item can be inserted before or after the currently selected list item (see Section 4.49 [SELECTITEM], page 45 command). The default is to insert them after the currently selected list item.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Enable result codes. */

OPTIONS RESULTS

/* Get a file name from the user. */

REQUESTFILE TITLE '"Select a file to upload"'

/* Add the file name to the upload list. */

IF rc = 0 THEN ADDITEM TO upload NAME result

/* Add phonebook entry #2 to the dial list. */

ADDITEM TO dial PHONE 2

/* Add all phonebook entries whose names start
 * with an `a' to the dial list.
 */

ADDITEM TO dial PHONE a#?

/* Add a plain phone number to the dial list. */

ADDITEM TO dial NAME 424242
```

4.3 The BAUD command

`Format:`

BAUD [Rate] <Baud rate in bits per second>

`Template:`

RATE/A/N

`Purpose:`

Sets the serial line transfer speed

`Specifications:`

Sets the serial line transfers speed to some defined value. The rate parameter passed in will be matched against all valid baud rates supported by 'term', the closest value will be used.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Change the serial transfer speed to 2400 bps. */

BAUD 2400
```

4.4 The BEEPSCREEN command

`Format:`

BEEPSCREEN

`Template:`

,

`Purpose:`

'Beeps' the terminal screen.

`Specifications:`

Invokes a bell signal, as configured in the program settings.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Invoke a bell signal. */

BEEPSCREEN
```

4.5 The CALLMENU command

`Format:`

CALLMENU [Title] <Title text or wildcard pattern>

`Template:`

TITLE/A/F

`Purpose:`

Invokes the function associated with a menu item.

`Specifications:`

Calls a pull-down menu function just as if the user had selected it using the mouse. The `Title` parameter can be any valid menu item name or a wildcard pattern. In the latter case, only the first menu item to match the pattern will be called.

`Result:`

-

`Warning:`

If no matching menu item was to be found.

`Example:`

```
/* Invoke the 'About...' menu item. */

CALLMENU abou#?
```

4.6 The CAPTURE command

`Format:`

CAPTURE [To] <Printer|File> [Name <File name>]

`Template:`

TO/A,NAME/K

`Purpose:`

Starts a file or printer capture.

`Specifications:`

If a capture is not already in progress will open a capture file or start capturing incoming terminal text to the printer. If the `File` argument is given and the `Name` parameter is omitted, will prompt for the name of a file to capture to.

If to capture to a given file, will append the captured text. If user is to select a file to capture to, will ask whether to append the text to the file or to overwrite it.

`Result:`

-

`Warning:`

In case user was to select a file and aborted the selection.

`Example:`

```
/* Open a named capture file. */

CAPTURE TO file NAME 'ram:capture file'

/* Close the capture file, ask the user for a file name. */

CLOSE FILE
CAPTURE TO file

/* Capture to the printer. */

CAPTURE TO printer
```

4.7 The CLEAR command

`Format:`

CLEAR [From] <Upload | Download | Dial | Wait | Buffer> [Force]

`Template:`

FROM/A,FORCE/S

`Purpose:`

Clears the contents of a global list or the text buffer.

`Specifications:`

This command serves to clear the contents of the lists to be maintained using the Section 4.2 [ADDITEM], page 11, Section 4.36 [REMITEM], page 36, Section 4.49 [SELECTITEM], page 45, etc. commands and to purge the contents of the text buffer. In the latter case the program will prompt for confirmation in case the buffer still holds any lines. This confirmation can be suppressed by using the `Force` parameter.

`Result:`

-

`Warning:`

In case the user did not confirm to clear the buffer.

`Example:`

```
/* Clear the wait list. */

CLEAR FROM wait

/* Clear the buffer, ask for a confirmation. */
```

```
CLEAR FROM buffer

/* If no confirmation was given, clear it by force. */

IF rc ~= 0 THEN CLEAR FROM buffer FORCE
```

4.8 The CLEARSCREEN command

`Format:`
: CLEARSCREEN

`Template:`
: ,

`Purpose:`
: Clears the terminal screen

`Specifications:`
: Clears the terminal screen and positions the cursor in the top left corner.

`Result:`
: -

`Warning:`
: -

`Example:`

```
/* Clear the terminal screen. */

CLEARSCREEN
```

4.9 The CLOSE command

`Format:`
: CLOSE [From] <Printer|File|All>

`Template:`
: FROM/A

`Purpose:`
: Terminates file and/or printer capture.

`Specifications:`
: Terminates a capture process as started with the Section 4.6 [CAPTURE], page 14 command.
Will terminate printer capture, file capture or both.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Terminate both file and printer capture. */

CLOSE ALL
```

4.10 The CLOSEDEVICE command

`Format:`

CLOSEDEVICE

`Template:`

,

`Purpose:`

Release the current serial device driver

`Specifications:`

Frees the serial device driver for use with other applications. The driver can be reopened (or a different device driver can be selected) using the Section 4.24 [OPENDEVICE], page 27 command.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Release the serial device driver, all serial I/O
 * will be halted.
 */

CLOSEDEVICE
```

4.11 The CLOSEREQUESTER command

`Format:`

CLOSEREQUESTER

Template:

,

Purpose:

Closes the currently open requester window

Specifications:

Will close any currently open requester window, such as the dialing window, the phone book, the serial settings window, etc. Will not close windows such as the file transfer window or the text/numeric input windows.

Result:

-

Warning:

-

Example:

```
/* Close the currently open requester window,
 * whatever it may be.
 */

CLOSEREQUESTER
```

4.12 The DEACTIVATE command

Format:

DEACTIVATE

Template:

,

Purpose:

Iconifies the program.

Specifications:

Puts the application to sleep. Requires Workbench to be running, so an AppIcon can be put on the Workbench backdrop. This command will be ignored if the application has already been put to sleep. To wake the application up, use the Section 4.1 [ACTIVATE], page 10 command.

Result:

-

Warning:

-

Example:

```
/* Iconify the program. */

DEACTIVATE
```

4.13 The DELAY command

`Format:`

DELAY [MIC | MICROSECONDS <Number>] [[SEC | SECONDS] <Number>] [MIN | MINUTES <Number>] [QUIET]

`Template:`

MIC=MICROSECONDS/K/N,SEC=SECONDS/N,MIN=MINUTES/K/N,QUIET/S

`Purpose:`

Delays program execution for a couple of microseconds, seconds and minutes.

`Specifications:`

Will cause both the program to make the call and the application to wait for a certain period of time. Unless the `QUIET` option is in effect will process and display data received from the serial line.

`Result:`

-

`Warning:`

If command was aborted before the timeout had elapsed.

`Example:`

```
/* Wait for five seconds. */

DELAY 5

/* Wait for one second and seven microseconds. */

DELAY MIC 7 SEC 5
```

4.14 The DIAL command

`Format:`

DIAL [WAIT | SYNC] [[Num] <Phone number>]

`Template:`

WAIT=SYNC/S,NUM/F

Purpose:

Dials the provided phone number. If no phone number was given, will dial the numbers and phone book entries stored in the dial list.

Specifications:

This command will build a dialing list from the available sources and pass it to the dialing function which is to schedule the dialing process and perform any login-actions. Available sources are the `Num` parameter which will cause the command to dial only this single number or the dial list whose contents will be used if the `Num` parameter is omitted.

If the `WAIT` parameter is used the command will wait until a connection is made. If the parameter is not use this command will return as soon as the dialing process has been initiated.

Result:

-

Warning:

If no connection was to be made.

Example:

```
/* Dial a single phone number. */

DIAL 424242

/* Wait a bit and abort the dialing process. */

DELAY 5
CLOSEREQUESTER

/* Clear the dialing list, then add all the phonebook entries
 * to the list.
 */

CLEAR FROM dial
ADDITEM TO dial PHONE #?

/* Dial the dial list. */

DIAL WAIT

/* Did we get a connection? */

IF RC == 0 THEN SEND TEXT "Ack!\r"
```

4.15 The DUPLEX command

Format:

DUPLEX [Full|Half|Echo]

`Template:`

FULL/S,HALF=ECHO/S

`Purpose:`

Sets the serial line duplex mode.

`Specifications:`

Sets the serial line duplex mode, this can be either full duplex or half duplex (local echo).

`Result:`

-

`Warning:`

-

`Example:`

```
/* Enable local terminal echo. */

DUPLEX ECHO
```

4.16 The EXECTOOL command

`Format:`

EXECTOOL [Console] [Async] [Port] [Command] <File name>

`Template:`

CONSOLE/S,ASYNC/S,PORT/S,COMMAND/A/F

`Purpose:`

Executes a program.

`Specifications:`

Will load and execute an AmigaDOS program. The `Console` parameter will cause an output file or window to be opened, the `Async` parameter will cause the command to return as soon as the execution process has been launched. The `Port` parameter will cause the current ARexx host port name to be passed to the tool on the command line.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Launch the ‘Dir’ command. */

EXECTOOL CONSOLE COMMAND ’dir c:’
```

4.17 The FAULT command

`Format:`

FAULT [Code] <Error code>

`Template:`

CODE/A/N

`Purpose:`

Returns the descriptive text associated with an error code as returned by 'term'.

`Specifications:`

'term' will return error codes in the `term.lasterror` variable. In order to get the descriptive text associated with an error code, use this command. All internal Rexx and AmigaDOS errors codes are supported as well as the error codes special to 'term'.

`Result:`

The error description associated with the error code.

`Warning:`

-

`Example:`

```
/* Enable command results. */

OPTIONS RESULTS

/* Get the text associated with error #1001. */

FAULT 1001

/* Output the result. */

SAY result
```

4.18 The GETATTR command

`Format:`

GETATTR [Object] <Name> [Field] <Name> [Stem <Name>] [Var <Name>]

`Template:`

OBJECT/A,FIELD,STEM/K,VAR/K

`Purpose:`

Obtains information on an application attribute.

Specifications:

Obtains information on an object, if possible will store it in the `result` variable. If a stem or simple variable name is given using the `Stem` or `Var` parameters will store the information in this variable. If no `Field` parameter is given, will store the entire object contents which requires that the `Stem` parameter is given. For list of valid attributes see the section entitled Section 5.23 [Attributes], page 73.

Result:

Returns information either in `result` variable or in the supplied `Stem` or `Var` variable.

Warning:

-

Example:

```
/* Enable command results. */

OPTIONS RESULTS

/* Query the name of the ARexx host we are communicating with. */

GETATTR OBJECT term FIELD arexx

/* Output the result. */

SAY result

/* Same as above, but using a different syntax. */

GETATTR term.arexx
SAY result

/* Store the entire contents of the phone book in a
 * stem variable.
 */

GETATTR phonebook STEM book

/* Output the phonebook contents. */

SAY 'phone book contains' book.count 'entries'

DO i = 0 TO book.count - 1
   SAY 'entry #' i

   SAY 'name    :' book.i.name
   SAY 'number  :' book.i.number
   SAY 'comment :' book.i.commenttext
   SAY 'username:' book.i.username
END i
```

4.19 The GETCLIP command

`Format:`

GETCLIP [Unit <Number>]

`Template:`

UNIT/K/N

`Purpose:`

Retrieves the contents of the clipboard.

`Specifications:`

Obtains the contents of the clipboard and returns it in the `result` variable. Will optionally read from the given clipboard unit, but uses the unit number selected in the application settings by default. *Note that a string returned can be up to 65,536 characters long!*

`Result:`

Contents of the clipboard if it contains any text.

`Warning:`

If clipboard does not contain any text.

`Example:`

```
/* Enable command results. */

OPTIONS RESULTS

/* Get the primary clipboard contents. */

GETCLIP

/* Output the contents. */

IF rc ~= 0 THEN
   SAY ’clipboard does not contain any text’
ELSE
   SAY result
```

4.20 The GOONLINE command

`Format:`

GOONLINE

`Template:`

,

`Purpose:`

Cause 'term' to go into online state.

`Specifications:`

After this command is processed 'term' will immediately go into online state. If the carrier signal is to be checked and no signal is present 'term' will drop into offline state right away.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Go into online state. */

GOONLINE
```

4.21 The HANGUP command

`Format:`

HANGUP

`Template:`

,

`Purpose:`

Hang up the serial line.

`Specifications:`

Hangs up the serial line, executes logoff and cleanup operations.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Hang up the line, whether the program is online or not. */

HANGUP
```

4.22 The HELP command

`Format:`

HELP [[Command] <Name>] [Prompt]

`Template:`

COMMAND,PROMPT/S

`Purpose:`

Returns the template of a command or invokes the online help system.

`Specifications:`

This command will either return the template associated with a 'term' ARexx command specified using the `Command` parameter or invoke the AmigaGuide(tm) help system.

`Result:`

Command template if requested.

`Warning:`

-

`Example:`

```
/* Enable command results. */

OPTIONS RESULTS

/* Query the template associated with the `selectitem' command. */

HELP selectitem

/* Output the result. */

SAY result

/* Invoke the online help. */

HELP PROMPT
```

4.23 The OPEN command

`Format:`

OPEN [Name <File name>] [TO] <Translations | Functionkeys | Cursorkeys | Fastmacros | Hotkeys | Speech | Sound | Buffer | Configuration | Phone>

`Template:`

NAME/K,TO/A

`Purpose:`

Reads data from a disk file.

`Specifications:`

This command reads the contents of a disk file and stores the information either in the configuration, the phone book or the text buffer. If text is read into the text buffer it will be appended to the existing text. If no file name is given will prompt the user to select a file.

`Result:`

-

`Warning:`

If user was requested to select a file and cancelled the selection.

`Example:`

```
/* Load the configuration from a file. */

OPEN NAME 'ram:term.prefs' TO configuration

/* Add text to the text buffer. */

OPEN TO buffer
```

4.24 The OPENDEVICE command

`Format:`

OPENDEVICE [Name <Device name>] [Unit <Number>]

`Template:`

NAME/K,UNIT/K/N

`Purpose:`

(Re-)Opens the serial device driver.

`Specifications:`

(Re-)Opens the previously released (see Section 4.10 [CLOSEDEVICE], page 17 command) device driver or a different device driver/unit if the corresponding `Device` and `Unit` parameters are given.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Release the serial device driver. */

CLOSEDEVICE
```

```
/* Open a different device driver. */

OPENDEVICE DEVICE 'duart.device' UNIT 5
```

4.25 The OPENREQUESTER command

`Format:`

OPENREQUESTER [REQUESTER] <Serial | Modem | Screen | Terminal | Emulation | Clipboard | Capture | Commands | Misc | Path | Transfer | Translations | Functionkeys | Cursorkeys | Fastmacros | Hotkeys | Speech | Sound | Phone>

`Template:`

REQUESTER/A

`Purpose:`

Opens a requester window.

`Specifications:`

Opens a requester window. Only a single window can be open at a time. The window opened can be closed using the Section 4.11 [CLOSEREQUESTER], page 17 command.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Open the phonebook window. */

OPENREQUESTER phone
```

4.26 The PARITY command

`Format:`

PARITY [Even | Odd | None | Mark | Space]

`Template:`

EVEN/S,ODD/S,NONE/S,MARK/S,SPACE/S

`Purpose:`

Sets the serial line parity mode.

`Specifications:`

Sets the serial line parity mode.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Set the parity mode. */

PARITY NONE
```

4.27 The PASTECLIP command

`Format:`

PASTECLIP [Unit <Number>]

`Template:`

UNIT/K/N

`Purpose:`

Feed the contents of the clipboard into the input stream.

`Specifications:`

Feeds the contents of the clipboard into the input stream. Obtains the data either from the given clipboard unit or from the default unit configured in the program settings.

`Result:`

-

`Warning:`

If clipboard does not contain any text.

`Example:`

```
/* Paste the contents of clipboard #2. */

PASTECLIP UNIT 2
```

4.28 The PRINT command

`Format:`

PRINT [From] <Screentext | Clipboard | Buffer | Dial | Wait | Upload | Download> [TO <File name>] [Serial | Modem | Screen | Terminal | User | Comment | Size | Date | Attr]

`Template:`

FROM/A,TO/K,SERIAL/S,MODEM/S,SCREEN/S,TERMINAL/S,USER/S, COMMENT/S,SIZE/S,DATE/S,ATTR/S

`Purpose:`

Prints the contents of the screen, the clipboard, the textbuffer or one of the lists.

`Specifications:`

Outputs the contents of the screen, the clipboard, the textbuffer or one of the lists (see Section 4.2 [ADDITEM], page 11 command) to a file or the printer. If the `To` parameter is omitted, will output the data to the printer. The parameters `Serial` through `Attr` control what will be printed:

`Screentext, Clipboard, Buffer, Wait list`

Options have no effect on the output.

`Dial list`

Responds to the `Serial`, `Modem`, `Screen`, `Terminal`, `User` and `Comment` parameters. The printout will contain information on the corresponding settings.

`Upload list, Download list`

Responds to the `Comment`, `Size`, `Date` and `Attr` parameters. The printout will contain information on the corresponding file attributes. *Note: if any of these parameters are given, only the base file names will be printed along with the corresponding information.*

`Result:`

-

`Warning:`

If user cancelled print operation.

`Example:`

```
/* Clear the dialing list, then add the entire phone book to it. */

CLEAR dial
additem to dial phone #?

/* Send the contents of the dial list to a disk file. */

PRINT FROM dial TO 'ram:phonebook' SERIAL MODEM SCREEN...
...TERMINAL USER COMMENT
```

4.29 The PROCESSIO command

`Format:`

PROCESSIO <On | Off>

`Template:`

ON/S,OFF/S

`Purpose:`

Turns serial I/O processing on or off.

`Specifications:`

Usually, the 'term' main program processes incoming data from the serial line, i.e. text is displayed in the terminal window or data transfers are started. This can interfere with custom I/O processing, such as done by an ARexx program which wants to receive and process all incoming serial data, without getting interrupted by the main program. For an application example see Section 4.59 [WAIT], page 51.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Turn off I/O processing. */

PROCESSIO OFF
```

4.30 The PROTOCOL command

`Format:`

PROTOCOL [None | RTSCTS | RTSCTSDTR]

`Template:`

NONE/S,RTSCTS/S,RTSCTSDTR/S

`Purpose:`

Sets the serial line handshaking protocol.

`Specifications:`

Sets the serial line handshaking protocol. See the main program documentation for details.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Set the handshaking protocol. */

PROTOCOL NONE
```

4.31 The PUTCLIP command

`Format:`

PUTCLIP [Unit <Unit>] [TEXT] <Text to store>

`Template:`

UNIT/K/N,TEXT/A/F

`Purpose:`

Stores text in the clipboard.

`Specifications:`

Stores the provided text in the clipboard. Will store it in the given clipboard unit if the `Unit` parameter is given. Will use the unit number configured in the program settings otherwise.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Store a short string in the clipboard. Note: can
 * only be up to 65,536 characters long.
 */

PUTCLIP 'hello sailor!'
```

4.32 The QUIT command

`Format:`

QUIT [Force]

`Template:`

FORCE/S

`Purpose:`

Terminates the application.

`Specifications:`

Terminates program execution, will ask for a confirmation to leave unless the `Force` parameter is used. *Caution: this command may fail if there are still output windows open on the 'term' screen.*

`Result:`

-

`Warning:`

If user did not confirm termination.

`Example:`

```
/* Try to terminate the program, ask for confirmation. */

QUIT

/* If no confirmation was given terminate by force. */

IF rc ~= 0 THEN QUIT FORCE
```

4.33 The READ command

`Format:`

READ [Num <Number of bytes>] [CR] [Noecho] [Verbatim] [[Prompt] <Prompt text>]

`Template:`

NUM/K/N,CR/S,NOECHO/S,VERBATIM/S,PROMPT/K/F

`Purpose:`

Reads a number of bytes or a string from the serial line.

`Specifications:`

If `Num` parameter is given will read a number of bytes from the serial line (*note: only a maximum of 65,536 bytes can be read*). The command will return when the read request has been satisfied, the timeout (settable using the Section 4.57 [TIMEOUT], page 50 command) has elapsed or the command was aborted.

If the `CR` parameter is given will handle simple line editing functions (`Backspace`, `Control-X`) and return a string as soon as the `Carriage return` key is pressed, the timeout (settable using the Section 4.57 [TIMEOUT], page 50 command) has elapsed or the command is aborted.

The `Noecho` parameter will cause 'term' not to echo typed characters back to the remote. *Note that in order to see any input on the local side the remote is to echo the characters typed back.*

If present, the `Prompt` text will be sent across the serial line, much the same as if it had been sent using the Section 4.50 [SEND], page 46 command.

This command pays attention to the current character translation table for incoming characters. If the characters are to be read without any changes made one has to use the `Verbatim` parameter.

`Result:`

The string read.

`Warning:`

If command was cancelled, no input was made or, if the `CR` parameter is used, the timeout elapsed.

`Example:`

```
/* Enable command results. */

OPTIONS RESULTS

/* Set the read timeout to five seconds. */

TIMEOUT 5

/* Read seven bytes. */

READ NUM 7

/* Output the result. */

IF rc ~= 0 THEN
   SAY 'no data was read'
ELSE
   SAY result

/* Turn the timeout off. */

TIMEOUT OFF

/* Prompt for input. */

READ CR PROMPT 'Enter a line of text:'

/* Output the result. */

IF rc ~= 0 THEN
   SAY 'no input was made'
ELSE
   SAY result
```

4.34 The RECEIVEFILE command

`Format:`

RECEIVEFILE [Mode <ASCII|Text|Binary>] [Name <File name>]

`Template:`

MODE/K,NAME/K

`Purpose:`

Receive one or more files using the XPR protocol.

`Specifications:`

Receives one or more files using the currently configured XPR protocol. The `Mode` parameter determines the file transfer mode (either plain ASCII, Text mode or binary file mode), if omitted the file(s) will be received in binary mode. Some file transfer protocols do not require any file names to be given as they have their own means to determine the names of the files to be received. However, a file name parameter can be given. If omitted the file transfer protocol will prompt the user for a file name if necessary.

The names of all files received are placed on the download list for processing. The list will be cleared before the transfer is started.

`Result:`

-

`Warning:`

If user cancelled file selection.

`Example:`

```
/* Start to receive a file in text mode. */

RECEIVEFILE MODE text
```

4.35 The REDIAL command

`Format:`

REDIAL

`Template:`

,

`Purpose:`

Redials the numbers remaining in the currently configured dialing list.

`Specifications:`

Redials the numbers which still remain in the dialing list built either by the phone book or by the Section 4.14 [DIAL], page 19 command. *Note that this command will return as soon as the dialing process is initiated.*

`Result:`

-

`Warning:`

If dialing list is empty.

Example:

```
/* Redial the list. */

REDIAL

/* Successful? */

IF rc ~= 0 THEN SAY 'dialing list is empty'
```

4.36 The REMITEM command

Format:

REMITEM [From] <Upload | Download | Dial | Wait> [Name <Name or wildcard>]

Template:

FROM/A,NAME/K/F

Purpose:

Removes one or more items from a list.

Specifications:

Removes one or more items from a list. If no `Name` parameter is given will remove the currently selected list item (selectable using the Section 4.49 [SELECTITEM], page 45 command). The `Name` parameter can be a proper name or a wildcard pattern.

Note: Cannot remove named items from the dial list.

Result:

-

Warning:

If no list item would match the name pattern.

Example:

```
/* Remove the currently selected item from the wait list. */

REMITEM FROM wait

/* Remove all items from the wait list which end with `z'. */

REMITEM FROM wait NAME '#?z'
```

4.37 The REQUESTFILE command

Format:

REQUESTFILE [Title <Title text>] [Path <Path name>] [File <File name>] [Pattern <Wildcard pattern>] [Multi] [Stem | Name <Variable name>]

`Template:`

TITLE/K,PATH/K,FILE/K,PATTERN/K,MULTI/S,STEM=NAME/K

`Purpose:`

Requests one or more file names from the user.

`Specifications:`

Requests one or more file names from the user. Will present a file requester with given title text and preset path, file name and pattern values. If only a single file name is to be requested will place the result in the `result` variable. The `Multi` parameter allows multiple files to be selected, the number of files selected and the file names will be placed in the variable specified using the `Stem` parameter.

`Result:`

The name of the file selected will be placed in the `result` variable. If multiple file were selected, will place the following information in the supplied stem variable:

`<Variable name>.COUNT`

The number of files selected.

`<Variable name>.0 through <Variable name>.n-1`

The file names selected.

`Warning:`

If user cancelled selection.

`Example:`

```
/* Enable command results. */

OPTIONS RESULTS

/* Request a single file name from the user. */

REQUESTFILE TITLE '"select a file"'

/* Output the result. */

IF rc ~= 0 THEN
   SAY 'no file was selected'
ELSE
   SAY result

/* Request several files. */

REQUESTFILE TITLE 'select several files' MULTI STEM names

/* Output the result. */
```

```
IF rc ~= 0 THEN
   SAY 'no files were selected'
ELSE DO
   SAY 'files selected:' names.count

   DO i = 0 TO names.count - 1
      SAY names.i
   END
END
```

4.38 The REQUESTNOTIFY command

`Format:`

REQUESTNOTIFY [Title <Title text>] [Prompt] <Prompt text>

`Template:`

TITLE/K,PROMPT/A/F

`Purpose:`

Notify the user with a message.

`Specifications:`

Opens a requester to notify the user, the prompt text can include line feed characters (`'0A'X`), the user will be able to answer the requester by clicking on a `Continue` button.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Notify the user. */

REQUESTNOTIFY TITLE '"Important information"' ...
...PROMPT 'This message is important'
```

4.39 The REQUESTNUMBER command

`Format:`

REQUESTNUMBER [Default <Default number>] [Prompt <Prompt text>]

`Template:`

DEFAULT/K/N,PROMPT/K/F

`Purpose:`
: Requests a numeric value from the user

`Specifications:`
: Requests a numeric value from the user, will display the provided prompt text or a default text and present the provided default number, so user can simply hit return to accept the defaults.

`Result:`
: The number the has entered.

`Warning:`
: If user cancelled requester.

`Example:`

```
/* Enable command results. */

OPTIONS RESULTS

/* Requester a single number. */

REQUESTNUMBER DEFAULT 42 PROMPT 'Enter the answer'

/* Output the result. */

IF rc ~= 0 THEN
   SAY 'no number was entered'
ELSE
   SAY result
```

4.40 The REQUESTRESPONSE command

`Format:`
: REQUESTRESPONSE [Title <Title text>] [Options <Options string>] [Prompt] <Prompt text>

`Template:`
: TITLE/K,OPTIONS/K,PROMPT/A/F

`Purpose:`
: Request a response from user.

`Specifications:`
: Requests a response from the user, uses provided title and prompt text and a number of options. If no options are specified will use `Yes|No` as the defaults.

`Result:`
: For `Options` passed as `Yes|Perhaps|No` will return 1 for `Yes`, 2 for `Perhaps` and return a warning for `No`.

`Warning:`

If user selected negative choice.

`Example:`

```
/* Enable command results. */

OPTIONS RESULTS

/* Request a response. */

REQUESTRESPONSE PROMPT 'Are you indecisive?' ...
...OPTIONS '"Yes|Don't know|No"'

/* Look at the response. */

IF rc ~= 0 THEN
   SAY 'Not indecisive'
ELSE DO
   IF result = 0 THEN
      SAY 'Indecisive'
   ELSE
      SAY 'Probably indecisive'
END
```

4.41 The REQUESTSTRING command

`Format:`

REQUESTSTRING [Secret] [Default <String>] [Prompt <Text>]

`Template:`

SECRET/S,DEFAULT/K,PROMPT/K/F

`Purpose:`

Requests a string from the user.

`Specifications:`

Requests a string from the user, will display the provided prompt text or a default text and present the provided default string, so user can simply hit return to accept the defaults.

If the `Secret` parameter is provided, will not display the characters typed.

`Result:`

The text the user entered.

`Warning:`

If user cancelled the requester.

`Example:`

```
/* Enable command results. */

OPTIONS RESULTS

/* Request a secret string. */

REQUESTSTRING SECRET DEFAULT '"hello sailor!"' ...
...PROMPT 'Enter secret message'

/* Output the result. */

IF rc ~= 0 THEN
   SAY 'no text was entered'
ELSE
   SAY result
```

4.42 The RESETSCREEN command

`Format:`

RESETSCREEN

`Template:`

,

`Purpose:`

Resets the terminal screen to defaults.

`Specifications:`

Resets the terminal screen to defaults, this includes clearing the screen, moving the cursor to the home position and resetting text, text rendering styles and colours.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Reset the terminal screen. */

RESETSCREEN
```

4.43 The RESETSTYLES command

`Format:`

RESETSTYLES

`Template:`

,

`Purpose:`

Resets the text rendering styles to defaults.

`Specifications:`

Resets the text rendering styles to defaults, turning off inverse video, boldface, italics, etc. modes.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Reset the text rendering styles. */

RESETSTYLES
```

4.44 The RESETTEXT command

`Format:`

RESETTEXT

`Template:`

,

`Purpose:`

Reset the terminal text to defaults.

`Specifications:`

Reset the terminal text to defaults, this includes switching back from graphics text or G1 mode.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Reset the terminal text. */

RESETTEXT
```

4.45 The RESETTIMER command

`Format:`
: RESETTIMER

`Template:`
: ,

`Purpose:`
: Reset the online timer.

`Specifications:`
: The online timer is reset to 00:00:00, regardless whether 'term' is currently online or not.

`Result:`
: -

`Warning:`
: -

`Example:`

```
/* Reset the online timer. */

RESETTIMER
```

4.46 The RX command

`Format:`
: RX [Console] [Async] [Command] <Command name>

`Template:`
: CONSOLE/S,ASYNC/S,COMMAND/A/F

`Purpose:`
: Invokes an ARexx macro file.

`Specifications:`
: Invokes an ARexx macro file, if `Console` argument specified opens a console output window, else uses '`NIL:`', if `Async` argument specified executes the macro asynchronously.

`Result:`
: -

`Warning:`
: -

`Example:`

```
/* Launch the ‘term’ command shell. */

RX CONSOLE ASYNC 'term:cmdshell.term'
```

4.47 The SAVE command

Format:

SAVE [From] <Translations|Functionkeys| Cursorkeys|Fastmacros|Hotkeys|Speech|Sound|Buffer|Configuration|Phone| Screentext|Screenimage>

Template:

FROM/A

Purpose:

Saves data to a disk file.

Specifications:

Saves data to a disk file, will prompt for a file name to save to. See Section 4.48 [SAVEAS], page 44 command for more information.

Result:

-

Warning:

If user cancels save operation.

Example:

```
/* Save the terminal screen contents to an
 * IFF-ILBM file.
 */

SAVE FROM screenimage
```

4.48 The SAVEAS command

Format:

SAVEAS [Name <File name>] [From] <Translations|Functionkeys|Cursorkeys| Fastmacros|Hotkeys|Speech|Sound|Buffer| Configuration|Phone|Screentext| Screenimage>

Template:

NAME/K,FROM/A

Purpose:

Saves data to a disk file.

`Specifications:`

Saves data to a disk file, will prompt for a filename to save to if none is provided. Will save either parts of the program configuration or the phone book contents (`Phonebook` parameter), the contents of the terminal screen as plain ASCII text (`Screentext` parameter) or the contents of the terminal screen as an IFF-ILBM-file (`Screenimage` parameter).

`Result:`

-

`Warning:`

If user cancels save operation.

`Example:`

```
/* Save the program configuration to a file. */

SAVEAS NAME 'ram:term.prefs' FROM configuration
```

4.49 The SELECTITEM command

`Format:`

SELECTITEM [Name <Name>] [From] <Upload | Download | Dial | Wait> [Next | Prev | Previous | Top | Bottom]

`Template:`

NAME/K,FROM/A,NEXT/S,PREV=PREVIOUS/S,TOP/S,BOTTOM/S

`Purpose:`

Select an item from a list.

`Specifications:`

Selects an item from a list, returns the item name in the `result` variable. The `Top` parameter will select the first list item, `Bottom` the last item. The `Previous` parameter will select the previous list item, `Next` the next successive item. Instead of using a positioning parameter, it is also possible to use a wildcard pattern or name with the `Name` parameter. The first list item to match the name will be selected.

Note: cannot be used with the dial list.

`Result:`

Returns the list item in the `result` variable.

`Warning:`

If end of list reached.

`Example:`

```
/* Enable command results. */

OPTIONS RESULTS
```

```
/* Output the contents of the download list. */

SELECTITEM FROM download TOP

DO WHILE rc = 0
   SAY result
   SELECTITEM FROM download NEXT
END
```

4.50 The SEND command

`Format:`

SEND [Noecho] [Local] [Literal] [Byte <ASCII code>] [Text] <Text>

`Template:`

NOECHO/S,LOCAL/S,LITERAL/S,BYTE/K/N,TEXT/A/F

`Purpose:`

Sends the provided text to the serial line, executes embedded command sequences.

`Specifications:`

Sends the provided text to the serial line, executes embedded command sequences (see main program documentation). To send a single byte, use the `Byte` parameter. The `Noecho` parameter will suppress terminal output. The `Local` parameter will cause the text to be output only locally in the terminal window, it will not be sent across the serial line. The `Literal` parameter keeps 'term' from interpreting any special characters, such as `\\r`, in the text to send and just transmits the text you passed in.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Send some text to the serial line. */

SEND 'This is some text.\r\n'

/* Send a single byte (a null) to the serial line. */

SEND BYTE 0

/* Execute an embedded command (send a break signal). */

SEND '\x'
```

4.51 The SENDBREAK command

`Format:`

SENDBREAK

`Template:`

,

`Purpose:`

Send a break signal across the serial line.

`Specifications:`

Send a break signal across the serial line.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Send a break signal. */

SENDBREAK
```

4.52 The SENDFILE command

`Format:`

SENDFILE [Mode <ASCII|Text|Binary>] [Names] {File names}

`Template:`

MODE/K,NAMES/M

`Purpose:`

Transfers files using the currently selected file transfer protocol.

`Specifications:`

Transfers one or more files using the currently configured XPR protocol. The `Mode` parameter determines the file transfer mode (either plain ASCII, Text mode or binary file mode), if omitted the file(s) will be sent in binary mode. Some file transfer protocols do not require any file names to be given as they have their own means to determine the names of the files to be sent. However, a file name parameter can be given. If omitted the file transfer protocol will prompt the user for a file name if necessary. Several file names can be given if necessary, they will be transferred along with the file names stored in the upload list. The file transfer process will remove any files successfully transferred from the upload list, leaving only those behing which were not to be transferred correctly.

Files whose names do not include a fully qualified path name are expected to reside in the default upload directory as specified in the main program paths settings.

`Result:`

-

`Warning:`

If user cancels file selection.

`Example:`

```
/* Prompt for files to be uploaded. */

SENDFILE

/* Send a single file. */

SENDFILE 'c:list'

/* Clear the upload list, add a single file name. */

CLEAR upload
ADDITEM TO upload NAME 'c:dir'

/* Transfer the file. */

SENDFILE
```

4.53 The SETATTR command

`Format:`

SETATTR [Object] <Name> [Field] <Name> [Stem <Name>] [Var <Name>]

`Template:`

OBJECT/A,FIELD,STEM/K,VAR

`Purpose:`

Sets a certain application attribute.

`Specifications:`

Sets a certain application attribute, retrieves the information from the supplied stem or simple variable. For a list of valid attributes, see the section entitled Section 5.23 [Attributes], page 73.

`Result:`

-

`Warning:`

-

Example:

```
/* Set the transfer speed. */

SETATTR serialprefs baudrate 2400
```

4.54 The SPEAK command

Format:
SPEAK [Text] <Text>

Template:
TEXT/A/F

Purpose:
Speaks the provided text using the Amiga speech synthesizer.

Specifications:
Speaks the provided text using the Amiga speech synthesizer, requires that speech support is enabled.

Result:
-

Warning:
-

Example:

```
/* Say something sensible. */

SPEAK 'something sensible'
```

4.55 The STOPBITS command

Format:
STOPBITS [0|1]

Template:
0/S,1/S

Purpose:
Sets the serial line stop bits.

Specifications:
Sets the serial line stop bits.

Result:

-

Warning:

-

Example:

```
/* Set the serial line stop bits. */

STOPBITS 1
```

4.56 The TEXTBUFFER command

Format:

TEXTBUFFER [Lock | Unlock]

Template:

LOCK/S,UNLOCK/S

Purpose:

Locks or unlocks the text buffer contents.

Specifications:

Locks or unlocks the text buffer contents, similar to the effect of the corresponding main menu entry.

Result:

-

Warning:

-

Example:

```
/* Lock the text buffer. */

TEXTBUFFER LOCK
```

4.57 The TIMEOUT command

Format:

TIMEOUT [[Sec | Seconds] <Number>] [Off]

Template:

SEC=SECONDS/N,OFF/S

`Purpose:`

Sets the serial read timeout.

`Specifications:`

Sets the timeout the Section 4.59 [WAIT], page 51 and Section 4.33 [READ], page 33 commands will wait until they exit.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Set the read timeout. */

TIMEOUT SEC 5
```

4.58 The TRAP command

`Format:`

TRAP <On | Off>

`Template:`

ON/S,OFF/S

`Purpose:`

Turns the trap list processing on or off.

`Specifications:`

This command tells the main program whether it should process entries of the trap list when filtering input or not.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Ignore the trap list. */

TRAP OFF
```

4.59 The WAIT command

`Format:`

WAIT [Noecho] [[Text] <Text>]

`Template:`

NOECHO/S,TEXT/F

`Purpose:`

Waits for a certain sequence of characters to be received from the serial line.

`Specifications:`

Wait for text to be received from the serial line. If no text to wait for is provided wait for either item of the wait list to appear. The `Noecho` parameter suppresses terminal output. *Note that text comparison does not consider the case of characters (in respect to the ECMA Latin 1 character set).* As 'term' has control over the incoming data stream before and after the `WAIT` command is executed, it may 'eat' and process data the `WAIT` command ought to receive. In order to avoid this effect you can use the `PROCESSIO` command (see Section 4.29 [PROCESSIO], page 30). For example, at the beginning of a program you could tell 'term' to leave the incoming data stream alone with the `PROCESSIO OFF` command, then invoke the `WAIT` command as needed, and eventually when your program exits allow 'term' to process incoming data with the `PROCESSIO ON` command.

`Result:`

Returns the string found.

`Warning:`

If timeout has elapsed before any matching text was received.

`Example:`

```
/* Enable command results. */

OPTIONS RESULTS

/* Set the read timeout. */

TIMEOUT SEC 30

/* Wait for a single line of text. */

WAIT 'some text'

/* Clear the wait list, add a few items. */

CLEAR wait
ADDITEM TO wait NAME 'foo'
ADDITEM TO wait NAME 'bar'

/* Wait for the text to appear. */

WAIT
```

```
/* Output the result. */

IF rc ~= 0 THEN
   SAY 'no text was received'
ELSE
   SAY result
```

4.60 The WINDOW command

`Format:`

WINDOW [Names] {<Buffer|Review|Packet|Fastmacros| Status|Main|UploadQueue>}
[Open|Close] [Activate] [Min|Max] [Front|Back] [Top|Bottom|Up|Down]

`Template:`

NAMES/A/M,OPEN/S,CLOSE/S,ACTIVATE/S,MIN/S,MAX/S,FRONT/S,BACK/S, TOP/S,BOTTOM/S,UP/S,DOWN/S

`Purpose:`

Manipulates the aspects of a window.

`Specifications:`

Manipulates the aspects of a window. Not all windows will support all available commands. The windows supported are:

`Buffer`

The text buffer window and screen. Supports the `Open`, `Close`, `Activate` and `Front` commands.

`Review`

The review window. Supports the `Open`, `Close`, `Activate`, `Min`, `Max`, `Front`, `Back`, `Top`, `Bottom`, `Up`, and `Down` commands.

`Packet`

The packet window. Supports the `Open`, `Close`, `Activate`, `Min`, `Max`, `Front` and `Back` commands.

`Fastmacros`

The fast! macro window. Supports the `Open`, `Close`, `Activate`, `Min`, `Max`, `Front` and `Back` commands.

`Status`

The status window. Supports the `Open`, `Close`, `Activate`, `Front` and `Back` commands.

`Main`

The main program window. Supports the `Open`, `Close`, `Activate`, `Front` and `Back` commands.

`Result:`

-

`Warning:`

-

`Example:`

```
/* Open all available windows. */

WINDOW buffer review packet fastmacros status main OPEN
```

5 Attributes

Several of the application's internal variables are can be accessed and modified using the Section 4.18 [GETATTR], page 22 and Section 4.53 [SETATTR], page 48 commands. Information is available on the objects and their associated fields explained below. Each line consists of the object and field name and the type of the available data:

`Numeric data`
: `<Object>.<Field>`
: Numeric

The information is a numeric value.

`Text data`
: `<Object>.<Field>`
: Text

The information is a text string.

`Boolean data`
: `<Object>.<Field>`
: Boolean

The information is a boolean value and can be `ON` or `OFF`.

`Mapped codes`
: `<Object>.<Field>`
: <Value 1> ... <Value n>

The information can be one of the given values.

5.1 The TERM object (read-only)

`TERM.VERSION`
: Text

The 'term' program revision.

`TERM.SCREEN`
: Text

The name of the public screen the 'term' main window has been opened on.

`TERM.SESSION.ONLINE`
: Boolean

Whether the program is currently online or not.

`TERM.SESSION.SESSIONSTART`
: Text

  Time and date when the 'term' program was started.

`TERM.SESSION.BYTESSENT`
: Numeric

`TERM.SESSION.BYTESRECEIVED`
: Numeric

`TERM.SESSION.CONNECTMESSAGE`
: Text

  The message issued by the modem when the connection was established.

`TERM.SESSION.BBSNAME`
: Text

`TERM.SESSION.BBSNUMBER`
: Text

`TERM.SESSION.BBSCOMMENT`
: Text

`TERM.SESSION.USERNAME`
: Text

`TERM.SESSION.ONLINEMINUTES`
: Numeric

  The number of minutes the program is currently connected to a BBS.

`TERM.SESSION.ONLINECOST`
: Numeric

  The cost of the connection to the BBS.

`TERM.AREXX`
: Text

  The name of the ARexx host port the program is communicating with.

`TERM.LASTERROR`
: Numeric

  The code corresponding to the error the last command has caused.

`TERM.TERMINAL.ROWS`
: Numeric

  The number of available terminal screen rows.

`TERM.TERMINAL.COLUMNS`
: Numeric

  The number of available terminal screen columns.

`TERM.BUFFER.SIZE`
: Numeric

  The size of the text buffer.

5.2 The PHONEBOOK object (read-only)

Available fields are:

`PHONEBOOK.COUNT`
: Numeric

  The number of entries in the phonebook. The single phonebook entries can be accessed as `PHONEBOOK.0...` through `PHONEBOOK.n-1...`.

`PHONEBOOK.n.NAME`
: Text

`PHONEBOOK.n.NUMBER`
: Text

`PHONEBOOK.n.COMMENTTEXT`
: Text

`PHONEBOOK.n.USERNAME`
: Text

`PHONEBOOK.n.PASSWORDTEXT`
: Text

5.3 The SERIALPREFS object

Available fields are:

`SERIALPREFS.BAUDRATE`
: Numeric

`SERIALPREFS.BREAKLENGTH`
: Numeric

  The break signal length in microseconds.

`SERIALPREFS.BUFFERSIZE`
: Numeric

`SERIALPREFS.DEVICENAME`
: Text

`SERIALPREFS.UNIT`
: Numeric

`SERIALPREFS.BITSPERCHAR`
: Numeric

The number of bits per transferred char. This can be either seven or eight.

`SERIALPREFS.PARITYMODE`
: `NONE EVEN ODD MARK SPACE`.

`SERIALPREFS.STOPBITS`
: Numeric

The number of stop bits to be used. This can be either 0 or 1.

`SERIALPREFS.HANDSHAKINGMODE`
: `NONE RTSCTS RTSCTSDSR`

`SERIALPREFS.DUPLEXMODE`
: `HALF FULL`

`SERIALPREFS.INTERNALXONXOFF`
: Boolean

`SERIALPREFS.HIGHSPEED`
: Boolean

`SERIALPREFS.SHARED`
: Boolean

`SERIALPREFS.STRIPBIT8`
: Boolean

`SERIALPREFS.CARRIERCHECK`
: Boolean

`SERIALPREFS.PASSXONXOFFTHROUGH`
: Boolean

`SERIALPREFS.QUANTUM`
: Numeric

`SERIALPREFS.USEOWNDEVUNIT`
: Boolean

`SERIALPREFS.OWNDEVUNITREQUEST`
: `RELEASE RELEASERETRY IGNORE`

5.4 The MODEMPREFS object

Available fields are:

`MODEMPREFS.MODEMINITTEXT`
: Text

`MODEMPREFS.MODEMEXITTEXT`
: Text

`MODEMPREFS.MODEMHANGUPTEXT`
: Text

`MODEMPREFS.DIALPREFIXTEXT`
: Text

`MODEMPREFS.DIALSUFFIXTEXT`
: Text

`MODEMPREFS.DIALMODE`
: `PULSE` `TONE`

`MODEMPREFS.NOCARRIERTEXT`
: Text

`MODEMPREFS.NODIALTONETEXT`
: Text

`MODEMPREFS.CONNECTTEXT`
: Text

`MODEMPREFS.VOICETEXT`
: Text

`MODEMPREFS.RINGTEXT`
: Text

`MODEMPREFS.BUSYTEXT`
: Text

`MODEMPREFS.OKTEXT`
: Text

`MODEMPREFS.ERRORTEXT`
: Text

`MODEMPREFS.REDIALDELAY`
: Numeric

  The redial delay in seconds

`MODEMPREFS.DIALRETRIES`
: Numeric

`MODEMPREFS.DIALTIMEOUT`
Numeric

The dial timeout in seconds

`MODEMPREFS.VERBOSEDIALING`
Boolean

`MODEMPREFS.CONNECTAUTOBAUD`
Boolean

`MODEMPREFS.HANGUPDROPSDTR`
Boolean

`MODEMPREFS.REDIALAFTERHANGUP`
Boolean

`MODEMPREFS.NOCARRIERISBUSY`
Boolean

`MODEMPREFS.CONNECTLIMIT`
Numeric

Time limit in minutes.

`MODEMPREFS.CONNECTLIMITMACRO`
Text

`MODEMPREFS.TIMETOCONNECT`
Numeric

5.5 The SCREENPREFS object

Available fields are:

`SCREENPREFS.COLOURMODE`
`TWO` `FOUR` `EIGHT` `SIXTEEN`

`SCREENPREFS.FONTNAME`
Text

`SCREENPREFS.FONTSIZE`
Numeric

`SCREENPREFS.MAKESCREENPUBLIC`
Boolean

`SCREENPREFS.SHANGHAIWINDOWS`
Boolean

`SCREENPREFS.BLINKING`
: Boolean

`SCREENPREFS.FASTERLAYOUT`
: Boolean

`SCREENPREFS.TITLEBAR`
: Boolean

`SCREENPREFS.STATUSLINEMODE`
: `DISABLED STANDARD COMPRESSED`

`SCREENPREFS.USEPUBSCREEN`
: Boolean

`SCREENPREFS.PUBSCREENNAME`
: Text

`SCREENPREFS.USEPENS`
: Boolean

`SCREENPREFS.WINDOWBORDER`
: Boolean

`SCREENPREFS.SPLITSTATUS`
: Boolean

`SCREENPREFS.ONLINEDISPLAY`
: `TIME COST BOTH`

5.6 The TERMINALPREFS object

Available fields are:

`TERMINALPREFS.BELLMODE`
: `NONE VISIBLE AUDIBLE BOTH SYSTEM`

`TERMINALPREFS.ALERTMODE`
: `NONE BELL SCREEN BOTH`

`TERMINALPREFS.EMULATIONMODE`
: `INTERNAL ATOMIC TTY EXTERNAL HEX`

`TERMINALPREFS.FONTMODE`
: `STANDARD IBM IBMRAW`

`TERMINALPREFS.SENDCRMODE`
: `IGNORE CR CRLF`

`TERMINALPREFS.SENDLFMODE`
: `IGNORE LF LFCR`

`TERMINALPREFS.RECEIVECRMODE`
: `IGNORE CR CRLF`

`TERMINALPREFS.RECEIVELFMODE`
: `IGNORE LF LFCR`

`TERMINALPREFS.NUMCOLUMNS`
: Numeric

`TERMINALPREFS.NUMLINES`
: Numeric

`TERMINALPREFS.KEYMAPNAME`
: Text

`TERMINALPREFS.EMULATIONNAME`
: Text

`TERMINALPREFS.FONTNAME`
: Text

`TERMINALPREFS.FONTSIZE`
: Numeric

`TERMINALPREFS.USETERMINALPROCESS`
: Boolean

5.7 The EMULATIONPREFS object

Available fields are:

`EMULATIONPREFS.CURSORMODE`
: `STANDARD APPLICATION`

`EMULATIONPREFS.NUMERICMODE`
: `STANDARD APPLICATION`

`EMULATIONPREFS.CURSORWRAP`
: Boolean

`EMULATIONPREFS.LINEWRAP`
: Boolean

`EMULATIONPREFS.INSERTMODE`
: Boolean

`EMULATIONPREFS.NEWLINEMODE`
: Boolean

`EMULATIONPREFS.FONTSCALEMODE`
: `NORMAL HALF`

`EMULATIONPREFS.SCROLLMODE`
: `JUMP SMOOTH`

`EMULATIONPREFS.DESTRUCTIVEBACKSPACE`
: `OFF OVERSTRIKE OVERSTRIKESHIFT`

`EMULATIONPREFS.SWAPBSDELETE`
: Boolean

`EMULATIONPREFS.PRINTERENABLED`
: Boolean

`EMULATIONPREFS.ANSWERBACKTEXT`
: Text

`EMULATIONPREFS.CLSRESETSCURSOR`
: Boolean

`EMULATIONPREFS.NUMPADLOCKED`
: Boolean

`EMULATIONPREFS.CURSORLOCKED`
: Boolean

`EMULATIONPREFS.FONTLOCKED`
: Boolean

`EMULATIONPREFS.WRAPLOCKED`
: Boolean

`EMULATIONPREFS.STYLELOCKED`
: Boolean

`EMULATIONPREFS.COLOURLOCKED`
: Boolean

`EMULATIONPREFS.MAXPRESCROLL`
: Numeric

`EMULATIONPREFS.MAXJUMP`
: Numeric

`EMULATIONPREFS.USEPENS`
: Boolean

5.8 The CLIPBOARDPREFS object

Available fields are:

`CLIPBOARDPREFS.UNIT`
: Numeric

`CLIPBOARDPREFS.CONVERTLF`
: Wahrheitswert

`CLIPBOARDPREFS.LINEDELAY`
: Numeric

  Paste line delay in 1/100 seconds.

`CLIPBOARDPREFS.CHARDELAY`
: Numeric

  Paste character delay in 1/100 seconds.

`CLIPBOARDPREFS.LINEPROMPTTEXT`
: Text

`CLIPBOARDPREFS.SENDTIMEOUT`
: Numeric

  Timeout in 1/100 seconds.

`CLIPBOARDPREFS.TEXTPACING`
: `DIRECT ECHO ANYECHO PROMPT DELAY KEYBOARD`

`CLIPBOARDPREFS.INSERTPREFIXTEXT`
: Text

`CLIPBOARDPREFS.INSERTSUFFIXTEXT`
: Text

5.9 The CAPTUREPREFS object

Available fields are:

`CAPTUREPREFS.LOGACTIONS`
: Boolean

`CAPTUREPREFS.LOGFILENAME`
: Text

`CAPTUREPREFS.LOGCALLS`
: Boolean

`CAPTUREPREFS.CALLLOGFILENAME`
: Text

`CAPTUREPREFS.MAXBUFFERSIZE`
: Numeric

`CAPTUREPREFS.BUFFER`
: Boolean

`CAPTUREPREFS.BUFFERSAVEPATH`
    Text

`CAPTUREPREFS.CONNECTAUTOCAPTURE`
    Boolean

`CAPTUREPREFS.AUTOCAPTUREDATE`
    `NAME`, `INCLUDE`

`CAPTUREPREFS.CAPTUREFILTER`
    Boolean

`CAPTUREPREFS.CONVERTCHARACTERS`
    Boolean

`CAPTUREPREFS.CAPTUREPATH`
    Text

`CAPTUREPREFS.OPENBUFFERWINDOW`
    `TOP`, `END`

`CAPTUREPREFS.REMEMBERBUFFERWINDOW`
    Boolean

`CAPTUREPREFS.OPENBUFFERSCREEN`
    `TOP`, `END`

`CAPTUREPREFS.REMEMBERBUFFERSCREEN`
    Boolean

`CAPTUREPREFS.BUFFERSCREENPOSITION`
    `LEFT`, `MID`, `RIGHT`

`CAPTUREPREFS.BUFFERWIDTH`
    Numeric

`CAPTUREPREFS.SEARCHHISTORY`
    Numeric

5.10 The COMMANDPREFS object

Available fields are:

`COMMANDPREFS.STARTUPMACROTEXT`
    Text

`COMMANDPREFS.LOGINMACROTEXT`
    Text

`COMMANDPREFS.LOGOFFMACROTEXT`
    Text

`COMMANDPREFS.UPLOADMACROTEXT`
Text

`COMMANDPREFS.DOWNLOADMACROTEXT`
Text

5.11 The MISCPREFS object

Available fields are:

`MISCPREFS.PRIORITY`
Numeric

`MISCPREFS.BACKUPCONFIG`
Boolean

`MISCPREFS.OPENFASTMACROPANEL`
Boolean

`MISCPREFS.RELEASEDEVICE`
Boolean

`MISCPREFS.CREATEICONS`
Boolean

`MISCPREFS.SIMPLEIO`
Boolean

`MISCPREFS.PROTECTIVEMODE`
Boolean

`MISCPREFS.IOBUFFERSIZE`
Numeric

5.12 The PATHPREFS object

Available fields are:

`PATHPREFS.ASCIIUPLOADPATH`
Text

`PATHPREFS.ASCIIDOWNLOADPATH`
Text

`PATHPREFS.TEXTUPLOADPATH`
: Text

`PATHPREFS.TEXTDOWNLOADPATH`
: Text

`PATHPREFS.BINARYUPLOADPATH`
: Text

`PATHPREFS.BINARYDOWNLOADPATH`
: Text

`PATHPREFS.CONFIGPATH`
: Text

`PATHPREFS.EDITORNAME`
: Text

`PATHPREFS.HELPFILENAME`
: Text

5.13 The TRANSFERPREFS object

Available fields are:

`TRANSFERPREFS.DEFAULTPROTOCOL`
: Text

`TRANSFERPREFS.ERRORNOTIFYCOUNT`
: Numeric

`TRANSFERPREFS.ERRORNOTIFYWHEN`
: `NEVER` `ALWAYS` `START` `END`

`TRANSFERPREFS.ASCIIUPLOADPROTOCOL`
: Text

`TRANSFERPREFS.ASCIIDOWNLOADPROTOCOL`
: Text

`TRANSFERPREFS.QUIETTRANSFER`
: Boolean

`TRANSFERPREFS.MANGLEFILENAMES`
: Boolean

`TRANSFERPREFS.LINEDELAY`
: Numeric

`TRANSFERPREFS.CHARDELAY`
: Numeric

`TRANSFERPREFS.LINEPROMPTTEXT`
: Text

`TRANSFERPREFS.TEXTPACING`
: `DIRECT` `ECHO` `ANYECHO` `PROMPT` `DELAY` `KEYBOARD`

`TRANSFERPREFS.SENDTIMEOUT`
: Numeric

`TRANSFERPREFS.STRIPBIT8`
: Boolean

`TRANSFERPREFS.IGNOREDATAPASTTERMINATOR`
: Boolean

`TRANSFERPREFS.TERMINATORCHAR`
: Numeric

`TRANSFERPREFS.TEXTUPLOADPROTOCOL`
: Text

`TRANSFERPREFS.TEXTDOWNLOADPROTOCOL`
: Text

`TRANSFERPREFS.BINARYUPLOADPROTOCOL`
: Text

`TRANSFERPREFS.BINARYDOWNLOADPROTOCOL`
: Text

`TRANSFERPREFS.OVERRIDEPATH`
: Boolean

`TRANSFERPREFS.SETARCHIVEDBIT`
: Boolean

`TRANSFERPREFS.COMMENTMODE`
: `IGNORE` `FILETYPE` `SOURCE`

`TRANSFERPREFS.TRANSFERICONS`
: Boolean

`TRANSFERPREFS.HIDEUPLOADICON`
: Boolean

`TRANSFERPREFS.TRANSFERPERFMETER`
: Boolean

`TRANSFERPREFS.DEFAULTTYPE`
: `XPR` or `PROGRAM`

`TRANSFERPREFS.DEFAULTSENDSIGNATURE`
: Text

`TRANSFERPREFS.DEFAULTRECEIVESIGNATURE`
: Text

`TRANSFERPREFS.ASCIIUPLOADTYPE`
: `XPR`, `PROGRAM`, `DEFAULT` or `INTERNAL`

`TRANSFERPREFS.ASCIIUPLOADSIGNATURE`
: Text

`TRANSFERPREFS.ASCIIDOWNLOADSIGNATURE`
: Text

`TRANSFERPREFS.ASCIIDOWNLOADTYPE`
: `XPR`, `PROGRAM`, `DEFAULT` or `INTERNAL`

`TRANSFERPREFS.ASCIIDOWNLOADSIGNATURE`
: Text

`TRANSFERPREFS.ASCIIDOWNLOADSIGNATURE`
: Text

`TRANSFERPREFS.TEXTUPLOADTYPE`
: `XPR`, `PROGRAM` or `DEFAULT`

`TRANSFERPREFS.TEXTUPLOADSIGNATURE`
: Text

`TRANSFERPREFS.TEXTDOWNLOADSIGNATURE`
: Text

`TRANSFERPREFS.TEXTDOWNLOADTYPE`
: `XPR`, `PROGRAM` or `DEFAULT`

`TRANSFERPREFS.TEXTDOWNLOADSIGNATURE`
: Text

`TRANSFERPREFS.TEXTDOWNLOADSIGNATURE`
: Text

`TRANSFERPREFS.BINARYUPLOADTYPE`
: `XPR`, `PROGRAM` or `DEFAULT`

`TRANSFERPREFS.BINARYUPLOADSIGNATURE`
: Text

`TRANSFERPREFS.BINARYDOWNLOADSIGNATURE`
: Text

`TRANSFERPREFS.BINARYDOWNLOADTYPE`
: `XPR`, `PROGRAM` or `DEFAULT`

`TRANSFERPREFS.BINARYDOWNLOADSIGNATURE`
: Text

`TRANSFERPREFS.BINARYDOWNLOADSIGNATURE`
: Text

5.14 The PROTOCOLPREFS object

This object features no fields, it contains a single line of text: the XPR protocol options.

5.15 The TRANSLATIONPREFS object

Indices referring to the ascii codes range from 0 to 255, available fields are:

`TRANSLATIONPREFS.n.SEND`
    Text

`TRANSLATIONPREFS.n.RECEIVE`
    Text

5.16 The FUNCTIONKEYPREFS object

Key indices range from 1 to 10 (representing F1 through F10), available fields are:

`FUNCTIONKEYPREFS.n`
    Text

`FUNCTIONKEYPREFS.SHIFT.n`
    Text

`FUNCTIONKEYPREFS.ALT.n`
    Text

`FUNCTIONKEYPREFS.CONTROL.n`
    Text

5.17 The CURSORKEYPREFS object

Available fields are:

`CURSORKEYPREFS.UPTEXT`
    Text

`CURSORKEYPREFS.RIGHTTEXT`
    Text

`CURSORKEYPREFS.DOWNTEXT`
: Text

`CURSORKEYPREFS.LEFTTEXT`
: Text

`CURSORKEYPREFS.SHIFT.UPTEXT`
: Text

`CURSORKEYPREFS.SHIFT.RIGHTTEXT`
: Text

`CURSORKEYPREFS.SHIFT.DOWNTEXT`
: Text

`CURSORKEYPREFS.SHIFT.LEFTTEXT`
: Text

`CURSORKEYPREFS.ALT.UPTEXT`
: Text

`CURSORKEYPREFS.ALT.RIGHTTEXT`
: Text

`CURSORKEYPREFS.ALT.DOWNTEXT`
: Text

`CURSORKEYPREFS.ALT.LEFTTEXT`
: Text

`CURSORKEYPREFS.CONTROL.UPTEXT`
: Text

`CURSORKEYPREFS.CONTROL.RIGHTTEXT`
: Text

`CURSORKEYPREFS.CONTROL.DOWNTEXT`
: Text

`CURSORKEYPREFS.CONTROL.LEFTTEXT`
: Text

5.18 The FASTMACROPREFS object

`FASTMACROPREFS.COUNT`
: Numeric

: The number of fast macros available, entries range from `FASTMACROPREFS.0...` to `FASTMACROPREFS.n-1...`

`FASTMACROPREFS.n.NAME`
: Text

`FASTMACROPREFS.n.CODE`
Text

5.19 The HOTKEYPREFS object

Available fields are:

`HOTKEYPREFS.TERMSCREENTOFRONTTEXT`
Text

`HOTKEYPREFS.BUFFERSCREENTOFRONTTEXT`
Text

`HOTKEYPREFS.SKIPDIALENTRYTEXT`
Text

`HOTKEYPREFS.ABORTAREXX`
Text

`HOTKEYPREFS.COMMODITYPRIORITY`
Numeric

`HOTKEYPREFS.HOTKEYSENABLED`
Boolean

5.20 The SPEECHPREFS object

Available fields are:

`SPEECHPREFS.RATE`
Numeric

`SPEECHPREFS.PITCH`
Numeric

`SPEECHPREFS.FREQUENCY`
Numeric

`SPEECHPREFS.SEXMODE`
`MALE FEMALE`

`SPEECHPREFS.VOLUME`
Numeric

`SPEECHPREFS.SPEECH`
Boolean

5.21 The SOUNDPREFS object

Available fields are:

`SOUNDPREFS.BELLNAME`
Text

`SOUNDPREFS.CONNECTNAME`
Text

`SOUNDPREFS.DISCONNECTNAME`
Text

`SOUNDPREFS.GOODTRANSFERNAME`
Text

`SOUNDPREFS.BADTRANSFERNAME`
Text

`SOUNDPREFS.RINGNAME`
Text

`SOUNDPREFS.VOICENAME`
Text

`SOUNDPREFS.ERRORNAME`
Text

`SOUNDPREFS.PRELOAD`
Boolean

5.22 The CONSOLEPREFS object

This object features no fields, it contains a single line of text: the console output window specification.

5.23 The FILEPREFS object

Available fields are:

`FILEPREFS.TRANSFERPROTOCOLNAME`
Text

`FILEPREFS.TRANSLATIONFILENAME`
Text

`FILEPREFS.MACROFILENAME`
: Text

`FILEPREFS.CURSORFILENAME`
: Text

`FILEPREFS.FASTMACROFILENAME`
: Text

6 Wanted!

As of this writing only a single example ARexx script is included in the ‘term’ distribution (see the ‘`Rexx`’ drawer). However, it is desirable to include more sample scripts so more users will be able to take advantage of the ARexx interface.

If you wish to share your scripts with the ‘term’ user community, send them (including documentation) to:

Olaf Barthel
Brabeckstrasse 35
D-30559 Hannover
Federal Republic of Germany
Internet: olsen@sourcery.han.de

Index

O

P

Q

R

S

T

U

W

Table of Contents