

**NAME**

picocom – minimal dumb–terminal emulation program

**SYNOPSIS**

**picocom** [ *options* ] *device*

**DESCRIPTION**

As its name suggests, **picocom(1)** is a minimal dumb–terminal emulation program. It is, in principle, very much like **minicom(1)**, only it's "pico" instead of "mini"! It was designed to serve as a simple, manual, modem configuration, testing, and debugging tool. It has also served (quite well) as a low–tech serial communications program to allow access to all types of devices that provide serial consoles. It could also prove useful in many other similar tasks.

In effect, picocom is not an "emulator" per–se. It is a simple program that opens, configures, manages a serial port (tty device) and its settings, and connects to it the terminal emulator you are, most likely, already using (the terminal window application, xterm, rxvt, system console, etc).

When picocom starts it opens the tty (serial port) given as its non–option argument. Unless the **--noinit** option is given, it configures the port to the settings specified by the option–arguments (or to some default settings), and sets it to "raw" mode. If **--noinit** is given, the initialization and configuration is skipped; the port is just opened. Following this, if standard input is a tty, picocom sets the tty to raw mode. Then it goes in a loop where it listens for input from stdin, or from the serial port. Input from the serial port is copied to the standard output while input from the standard input is copied to the serial port. Picocom also scans its input stream for a user–specified control character, called the *escape character* (being by default **C–a**). If the escape character is seen, then instead of sending it to the serial–device, the program enters "command mode" and waits for the next character (which is called the "function character"). Depending on the value of the function character, picocom performs one of the operations described in the **COMMANDS** section below.

**COMMANDS**

Commands are given to picocom by first keying the *espace character* which by default is **C–a** (see **OPTIONS** below for how to change it), and then keying one of the function (command) characters shown here.

*escape character*

Send the escape character to the serial port and return to "transparent" mode. This means that if the escape character (**C–a**, by default) is typed twice, the program sends the escape character to the serial port, and remains in transparent mode.

**C–x** Exit the program. If the **--noreset** option is *not* given, then the serial port is reset to its original settings before exiting, and the modem control lines (typically DTR and RTS) are cleared (lowered) signaling a modem hangup. If **--noreset** is given (and **--hangup** is not), then the serial port settings are not reset, and the modem control lines remain unaffected. If both **--noreset** and **--hangup** are given, then the serial port settings are not reset, but the modem–control lines *are* cleared.

**C–q** Quit the program *without* resetting the serial port to its original settings. Terminating with the Quit command, picocom behaves *exactly* as if the **--noreset** option was given. The serial port is *not* reset to its original settings, and the modem control lines remain unaffected or are cleared, subject to the **--hangup** option.

**C–p** Pulse the DTR line. Lower it for 1 sec, and then raise it again.

**C–t** Toggle the DTR line. If DTR is up, then lower it. If it is down, then raise it. May not be supported on some systems.

**C–g** Toggle the RTS line. If RTS is up, then lower it. If it is down, then raise it. Not supported if the flow control mode is RTS/CTS. May not be supported on some systems.

**C–backslash**

Generate a break sequence on the serial line. A break sequence is usually generated by marking (driving to logical one) the serial Tx line for an amount of time corresponding to several character

durations.

- C-b** Set baudrate. Prompts you to enter a baudrate numerically (in bps) and configures the serial port accordingly.
- C-u** Baud up. Increase the baud-rate. The list of baud-rates stepped-through by this command is: 50, 75, 110, 134, 150, 200, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200. If `HIGH_BAUD` support is compiled-in, then the following baud-rates are also added to the list: 230400, 460800, 500000, 576000, 921600, 1000000, 1152000, 1500000, 2000000, 2500000, 3000000, 3500000, 4000000. Depending on you system, any of the higher baud rates may be missing.
- C-d** Baud down. Decrease the baud-rate. The list of baud-rates stepped-through by this command is the same as for the "baud-up" command.
- C-f** Cycle through flow-control settings (RTS/CTS, XON/XOFF, none).
- C-y** Cycle through parity settings (even, odd, none).
- C-i** Cycle through databits-number settings (5, 6, 7, 8).
- C-j** Cycle through stopbits-number settings (1, 2).
- C-c** Toggle local-echo mode.
- C-w** Write hex. Picocom prompts the user for a string of hexadecimal values. Values can be entered with or without delimiters (separators). The hexadecimal values are translated to binary and sent to the port, exactly as if input at the terminal (i.e. the `--omap`, `--echo` and `--emap` options are observed). Example: The following sends the characters "ABCD" to the port.

```
C-a C-w
*** hex: 41 4243:44
*** wrote 4 bytes ***
```

- C-s** Send (upload) a file. See **SENDING AND RECEIVING FILES** below.
- C-r** Receive (download) a file. See **SENDING AND RECEIVING FILES** below.
- C-v** Show program options (like baud rate, data bits, etc) as well as the actual serial port settings. Only the options and port settings that can be modified online (through commands) are shown, not those that can only be set at the command-line. See **DISPLAY OF OPTIONS AND PORT SETTINGS** for details.

#### **C-h or C-k**

Show help, or show keys. Prints a short description of all available function (command) keys.

After performing one of the above operations, the program leaves the command mode and enters transparent mode. Example: To increase the baud-rate by two steps, you have to type:

**C-a, C-u, C-a, C-u**

assuming of-course that **C-a** is the escape character.

## **OPTIONS**

Picocom accepts the following command-line options.

#### **--baud | -b**

Defines the baud-rate to set the serial-port (terminal) to.

#### **--flow | -f**

Defines the flow-control mode to set the serial-port to. Must be one of: **x** for xon/xoff (software) mode, **h** for hardware flow control (RTS/CTS), **n** for no flow control. (Default: **n**)

#### **--parity | -y**

Defines the parity mode to set the serial-port to. Must be one of: **o** for odd parity mode, **e** for even parity mode, **n** for no parity mode. (Default: **n**)

**--databits | -d**

Defines the number of data bits in every character. Must be one of: **5**, **6**, **7**, **8**. (Default: **8**)

**--stopbits | -p**

Defines the number of stop bits in every character. Must be one of: **1**, or **2**. (Default: **1**)

**--escape | -e**

Defines the character that will make picocom enter command-mode (see description above). If **x** is given, then **C-x** will make picocom enter command mode. See also the **--no-escape** option. (Default: **a**)

**--no-escape | -n**

Disables the escape character. Picocom will never enter command-mode if this option is given. To exit picocom, in this case, you must either close its standard input, or send it the TERM or INT signal. (Default: Disabled).

**--echo | -c**

Enable local echo. Every character being read from the terminal (standard input) is echoed to the terminal (standard output) subject to the echo-mapping configuration (see **--emap** option). (Default: Disabled)

**--noinit | -i**

If given, picocom will not initialize, configure, or otherwise mess with the serial port at start-up. It will just open it. This is useful, for example, for connecting picocom to already-connected modems, or already configured ports without terminating the connection, or altering their settings. If required, serial port parameters can then be adjusted at run-time by commands. See also the **--noreset** and **--hangup** options. (Default: Disabled)

**--noreset | -r**

If given, picocom will not reset the serial port when exiting. It will just close the respective file descriptor and do nothing more. The serial port settings will *not* be restored to their original values and, unless the **--hangup** option is also given, the modem-control lines will *not* be affected. This is useful, for example, for leaving modems connected when exiting picocom. Regardless whether the **--noreset** option is given, the user can exit picocom using the "Quit" command (instead of "Exit"), which makes picocom behave *exactly* as if **--noreset** was given. See also the **--hangup** option. (Default: Disabled)

NOTICE: Picocom clears the modem control lines on exit by setting the *HUPCL* control bit of the respective port. Picocom always sets HUPCL according to the **--noreset** and **--hangup** options. If **--noreset** is given and **--hangup** is not, then HUPCL for the port is cleared and will remain so after exiting picocom. If **--noreset** is *not* given, or if both **--noreset** and **--hangup** are given, then HUPCL is set for the port and will remain so after exiting picocom. This is true, regardless of the way picocom terminates (command, read zero-bytes from standard input, killed by signal, fatal error, etc), and regardless of the **--noinit** option.

**--hangup | -u**

If given together with **--noreset**, picocom will not reset the serial port to its original settings on exit, but it *will* clear the modem control lines (typically DTR and RTS) to signal a modem hangup. Without the **--noreset** option (explicitly given, or implied by exiting with the "Quit" command) **--hangup** has no effect (without **--noreset** picocom always clears the modem control lines on exit, anyway).

**--nolock | -l**

If given, picocom will *not* attempt to lock the serial port before opening it. Normally, depending on how it's compiled, picocom attempts to get a UUCP-style lock-file (e.g. `/var/lock/LCK..ttyS0`) before opening the port, or attempts to lock the port device-node using **flock(2)**. Failing to do so, results in the program exiting after emitting an error-message. It is possible that your picocom binary is compiled without support for locking. In this case the **--nolock** option is accepted, but has no effect. (Default: Disabled)

**--send-cmd | -s**

Specifies the external program (and any arguments to it) that will be used for transmitting files. If the argument to **--send-cmd** is the empty string ("), the send-file command is disabled. See **SENDING AND RECEIVING FILES**. (Default: **sz -vv**)

**--receive-cmd | -v**

Specifies the external program (and any arguments to it) that will be used for receiving files. If the argument to **--receive-cmd** is the empty string ("), the receive-file command is disabled. See **SENDING AND RECEIVING FILES**. (Default: **rz -vv**)

**--imap**

Specifies the input character map (i.e. special characters to be replaced when read from the serial port). See **INPUT, OUTPUT, AND ECHO MAPPING**. (Default: Empty)

**--omap**

Specifies the output character map (i.e. special characters to be replaced before being written to serial port). See **INPUT, OUTPUT, AND ECHO MAPPING**. (Default: Empty)

**--emap**

Specifies the local-echo character map (i.e. special characters to be replaced before being echoed-back to the terminal, if local-echo is enabled). See **INPUT, OUTPUT, AND ECHO MAPPING**. (Default: **delbs,crcrlf**)

**--logfile | -g**

Use specified file for logging (recording) serial input, and possibly serial output. If the file exists, it is appended to. Every character read from the serial port is written to the specified file (before input mapping is performed). If local-echo mode is enabled (see **--echo** option and **C-c** command), then every character written to the serial port (after output mapping is performed) is also logged to the same file. (Default: no logging)

**--initstring | -t**

Send the provided string after opening and configuring the serial port. The init string is sent exactly as if it was input at the terminal. Sending the init string, picocom observes the **--omap** output mapping, the **--echo** local-echo setting, and the **--emap** local-echo mapping. This feature is useful, for example, if the serial device needs some special magic strings to start responding. Use **echo(1)** or **xxd(1)** to generate special characters like a CR or binary data. Example:

```
picocom -t "$(echo -ne 'AAATZ\r\n')" /dev/ttyS0
```

Note, that the init string is not sent if **--noinit** is given. (Default: empty).

**--lower-rts**

Lower the RTS modem control signal after opening the serial port. Only supported when flow-control mode is not set to RTS/CTS, ignored otherwise. Only supported on some systems.

If neither **--lower-rts** nor **--raise-rts** are given, the state of the RTS signal, after opening and configuring the port, is system dependent. On most systems the signal is raised.

**--raise-rts**

Raise the RTS modem control signal after opening the serial port. Only supported when flow-control mode is not set to RTS/CTS, ignored otherwise. Only supported on some systems.

If neither **--raise-rts** nor **--lower-rts** are given, the state of the RTS signal, after opening and configuring the port, is system dependent. On most systems the signal is raised.

**--lower-dtr**

Lower the DTR control signal after opening the serial port. Only supported on some systems.

If neither **--lower-dtr** nor **--raise-dtr** are given, the state of the DTR signal, after opening and configuring the port, is system dependent. On most systems the signal is raised.

**--raise-dtr**

Raise the DTR control signal after opening the serial port. Only supported on some systems.

If neither **--raise-dtr** nor **--lower-dtr** are given, the state of the DTR signal, after opening and configuring the port, is system dependent. On most systems the signal is raised.

**--exit-after | -x**

Exit picocom if it remains idle for the specified time (in milliseconds). Picocom is considered idle if: Nothing is read (received) from the serial port, AND there is nothing to write (send) to the serial port, AND nothing is read from the standard input (terminal). If **--exit-after** is set to zero, then picocom exits after opening and configuring the serial port, after sending the init string (if any, see option **--initstring**) and immediately when it becomes idle. When exiting after being idle, picocom drains the O/S serial port output buffer (i.e. waits for data already written to the port to be transmitted) and observes the **--noreset** and **--hangup** options as usual. (Default: not set).

NOTICE: If **--exit-after** is set, reading zero bytes from the standard input (which usually means that whatever was connected there has been closed), will *not* cause picocom to exit. Instead, picocom will keep running, *without* reading from stdin, and will exit only when it becomes idle for the specified time, or if it is killed by a signal. If **--exit-after** is *not* set, then reading zero bytes from the standard input causes picocom to exit, after the contents of its output queue have been transmitted.

**--exit | -X**

Exit picocom immediately after opening and configuring the serial port. Do *not* read *anything* from the standard input or from the serial port. When exiting the **--noreset** and **--hangup** options are observed as usual. With **--exit** and **--noreset** (and possibly **--hangup**) picocom can be used as a very crude replacement of **stty(1)**. If an init string is also given (see **--initstring** option), picocom exits immediately after sending (writing) the init string to the serial port and draining the O/S serial port output buffer (i.e. waiting for data written to the port to be transmitted). Again, nothing is read from the standard input, or from the serial port. The **--exit** option, overrides the **--exit-after** option. (Default: Disabled)

**--quiet | -q**

Forces picocom to be quiet. Suppresses the output of the initial status and options information, as well as any other information or messages not explicitly requested by the user. Responses to user commands and any error or warning messages are still printed.

**--help | -h**

Print a short help message describing the command-line options. Picocom's version, compile-time options, and enabled features are also shown.

**DISPLAY OF OPTIONS AND PORT SETTINGS**

The "show program options" command (**C-v**), as well as the commands that change program options (**C-b**, **C-u**, **C-d**, **C-f**, etc) print messages showing the current values (or the new values, if they were changed) for the respective options. If picocom determines that an actual serial-port setting differs from the current value of the respective option (for whatever reason), then the value of the option is shown followed by the value of the actual serial-port setting in parenthesis. Example:

```
*** baud: 115200 (9600)
```

This means that a baud rate of 115200bps has been selected (from the command line, or using commands that change the baudrate) but the serial-port is actually operating at 9600bps (the driver may not support the higher setting, and has silently replaced it with a safe default, or the setting may have been changed from outside picocom). If the option and the corresponding serial-port setting are the same, only a single value is shown. Example:

```
*** baud: 9600
```

This behavior was introduced in picocom 2.0. Older releases displayed only the option values, not the actual serial-port settings corresponding to them.

On startup, after the serial port is opened and configured (and assuming that neither the **--noinit**, nor the

—**quiet** command line options have been given), the port settings are silently checked. If any mismatch is detected between the requested and the actual port settings, a warning message is displayed. You may then use the **C-v** command to determine the exact mismatch or mismatches.

## SENDING AND RECEIVING FILES

Picocom can send and receive files over the serial port using external programs that implement the respective protocols. In Linux typical programs for this purpose are:

- **rx(1)** – receive using the X-MODEM protocol
- **rb(1)** – receive using the Y-MODEM protocol
- **rz(1)** – receive using the Z-MODEM protocol
- **sx(1)** – send using the X-MODEM protocol
- **sb(1)** – send using the Y-MODEM protocol
- **sz(1)** – send using the Z-MODEM protocol
- **ascii-xfr(1)** – receive or transmit ASCII files

The name of, and the command-line options to, the program to be used for transmitting files are given by the **--send-cmd** option. Similarly the program to receive files, and its arguments, are given by the **--receive-cmd** option. For example, in order to start a picocom session that uses **sz(1)** to transmit files, and **rz(1)** to receive files, you have to say something like this:

```
picocom --send-cmd "sz -vv" --receive-cmd "rz -vv" ...
```

If the argument to the **--send-cmd** option, or the argument to the **--receive-cmd** option is the empty string, then the respective command is disabled. For example, in order to disable both the "send" and the "receive" commands you can invoke picocom like this:

```
picocom --send-cmd '' --receive-cmd '' ...
```

A picocom session with both, the send- and the receive-file commands disabled does not **fork(2)** and does not run any external programs.

During the picocom session, if you key the "send" or "receive" commands (e.g. by pressing **C-a**, **C-s**, or **C-a**, **C-r**) you will be prompted for a filename. At this prompt you can enter one or more file-names, and any additional arguments to the transmission or reception program. Command-line editing and rudimentary pathname completion are available at this prompt, if you have compiled picocom with support for the linenoise library. Pressing **C-c** at this prompt will cancel the file transfer command and return to normal picocom operation. After entering a filename (and / or additional transmission or reception program arguments) and assuming you have not canceled the operation by pressing **C-c**, picocom will start the external program as specified by the **--send-cmd**, or **--receive-cmd** option, and with any filenames and additional arguments you may have supplied. The standard input and output of the external program will be connected to the serial port. The standard error of the external program will be connected to the terminal which—while the program is running—will revert to canonical mode. Pressing **C-c** while the external program is running will prematurely terminate it (assuming that the program itself does not ignore SIGINT), and return control to picocom. Pressing **C-c** at any other time, has no special effect; the character is normally passed to the serial port.

## INPUT, OUTPUT, AND ECHO MAPPING

Using the **--imap**, **--omap**, and **--emap** options you can make picocom map (translate, replace) certain special characters after being read from the serial port (with **--imap**), before being written to the serial port (with **--omap**), and before being locally echoed to the terminal (standard output) if local echo is enabled (with **--emap**). These mapping options take, each, a single argument which is a comma-separated list of one or more of the following identifiers:

- **crlf** (map CR to LF),
- **crclrf** (map CR to CR + LF),

- **igncr** (ignore CR),
- **lfer** (map LF to CR),
- **lfcrLf** (map LF to CR + LF),
- **ignlf** (ignore LF),
- **bsdel** (map BS to DEL),
- **delbs** (map DEL to BS)
- **spchex** (map special chars (< 0x20 || 0x7f), excl. CR, LF, and TAB to hex)
- **tabhex** (map TAB to hex)
- **crhex** (map CR to hex)
- **lfhex** (map LF to hex)
- **8bithex** (map chars with 8th-bit set to hex)
- **nrmhex** (map normal ascii chars (0x20 <= c < 0x7f) to hex)

The "to hex" mappings (**???hex**) replace the respective characters with their hexadecimal representation (in square brackets), like this:

```
CR --> [0d]
```

If more than one mappings are provided that apply to the same character, then only the first mapping, in the order listed above, is applied.

For example the command:

```
picocom --omap crlf,delbs --imap ignlf,bsdel --emap crcrlf ...
```

will:

- Replace every CR (carriage return, 0x0d) character with LF (line feed, 0x0a) and every DEL (delete, 0x7f) character with BS (backspace, 0x08) before writing it to the serial port.
- Ignore (not write to the terminal) every LF character read from the serial port, and replace every BS character read from the serial port with DEL.
- Replace every CR character with CR and LF when echoing to the terminal (if local-echo is enabled).

## EXITING PICOCOM

This section summarizes the conditions in which picocom terminates its operation and what happens in each such condition:

- The exit command is seen in the standard input. That is, the escape character is seen (default **C-a**), followed by the exit command character (default **C-x**). In this case: The contents of the output queue (data read from the standard input, but not yet written to the port) as well as the contents of the O/S serial port output buffer (data already written to the port, but not yet transmitted) are discarded (flushed). Then the serial port is reset to its original settings, and the modem-control lines are cleared signaling a modem reset, subject to the **--noreset** and the **--hangup** options. After that picocom exits with a success status.
- The quit command is seen in the standard input. That is, the escape character is seen (default **C-a**), followed by the quit command character (default **C-q**). The behavior in this case is similar to that of the exit command, with one difference: Picocom behaves as if the **--noreset** option is given (regardless if it actually is, or not).
- The **--exit** option is given. See the documentation of this option for a description of what exactly happens in this case. Picocom exits with a success exit status.
- The **--exit-after** option is given. See the documentation of this option for a description of what exactly happens in this case. Picocom exits with a success exit status.

- Zero bytes are read from the standard input. This usually means that whatever was connected to picocom's standard input has been closed or, if a file was connected, then picocom has read up to the end of the file. In this case, if the **--exit-after** option is *not* given, picocom stops reading from the standard input, and keeps operating normally (i.e. writing to, and reading from, the serial port) until its output queue empties. When this happens, picocom waits for the O/S serial port output buffer to drain and then (subject to the **--noreset** and **--hangup** options) resets the serial port to its initial settings, clears the modem-control lines, and exits. If the **--exit-after** option is given then, again, picocom stops reading from the standard input and continues operating normally but, in this case, it does so until it becomes idle for the specified amount of time, before exiting. Picocom exits with a success exit status.
- Picocom is killed by the TERM or INT signal, or an unrecoverable error occurs. In this case picocom behaves as if it had received the exit command, that is: The contents of the output queue and the contents of the O/S serial port output buffer are discarded (flushed). Then, subject to the **--noreset** and **--hangup** options, the serial port is reset to its original settings, the modem control lines are cleared, and picocom exits with a failure status.

## AUTHOR

Written by Nick Patavalis <npat@efault.net>

## AVAILABILITY

Download the latest release from: <<https://github.com/npat-efault/picocom/releases>>

## COPYRIGHT

Copyright (c) 2003–2018 Nick Patavalis

This file is part of Picocom.

Picocom is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

Picocom is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111–1307 USA