RPi Serial Connection

From eLinux.org

Back to RPi Advanced Setup.

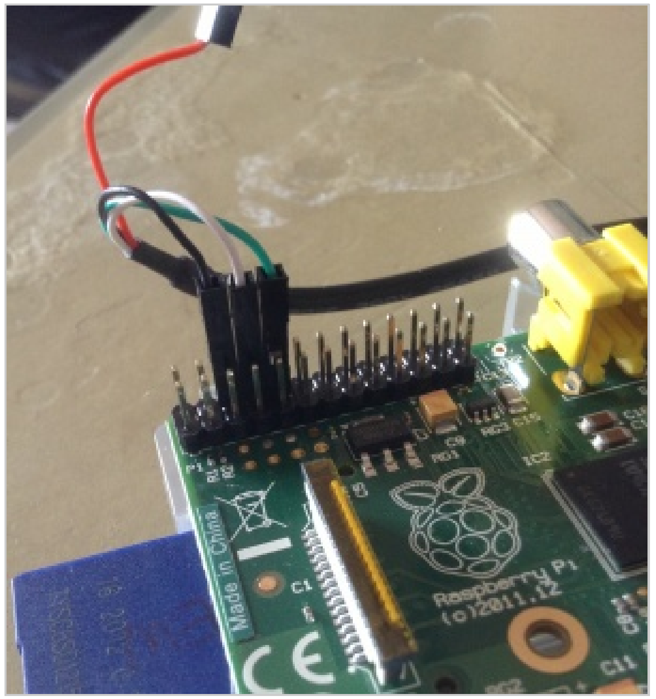
The Serial Port is a low-level way to send data between the Raspberry Pi and another computer system. There are two main ways in which it can be used:

- Connecting to a PC to allow access to the Linux console (http://en.wikipedia.org/wiki/System_console). This can help to fix problems during boot, or to log in to the Pi if the video and network are not available.
- Connecting to a microcontroller or other peripheral which has a serial (http://en.wikipedia.org/wiki/Serial_port) interface. This can be useful if you want the Pi to control another device.

Contents

Connections and signal levels

The Raspberry Pi serial port consists of two signals (a 'transmit' signal, TxD and a 'receive' signal RxD) made available on the GPIO header. To connect to another serial device, you connect the 'transmit' of one to the 'receive' of the other, and vice versa. You will also need to connect the Ground pins of the two devices together.


Adafruit serial cable connected to Pi

The Broadcom chip at the heart of the Pi uses 0 and 3.3V logic levels, not the +/-12V used by RS-232 (http://en.wikipedia.org/wiki/RS-232) serial ports found on some older PCs. If you wish to connect one of these, you need a board or adapter to convert the signal levels. See this tutorial (http://codeandlife.com/2012/07/01/raspberry-pi-serial-console-with-max3232cpe/) for one example on how to build a 3.3V to RS-232 level converter with a breadboard, a MAX3232CPE IC and five 0.1 uF capacitors.

If you wish to connect your Pi to a PC with a USB port, the simplest option is to use a USB-to-serial cable which uses 3.3V logic levels (e.g. the Adafruit 954 (http://www.adafruit.com/products/954) or the FTDI TTL-232R-RPI (http://www.ftdichip.com/Products/Cables/RPi.htm) cables). These can be simply plugged in directly to the GPIO header (see illustration).

When using the Adafruit 954 USB cable on Windows 7 the Prolific driver seems already be installed, so you could skip this step. Simply connect the USB cable. You will see a device Prolific USB-to-Serial Comm Port; verify via Control Panel → System → Device Manager to get the port number.

If you wish to connect to a peripheral which has 0/5V signals, you should ideally have a circuit to convert between the voltage levels. See this tutorial (http://www.element14.com/community/groups/raspberry-pi/blog/2012/07/18/look-ma-no-display-using-the-raspberry-pi-serial-console) for an example using a ready-made level shifter module. Other circuits for level shifting are shown at RPi_GPIO_Interface_Circuits#Level_Shifters.

Connection to a PC

You can connect the Pi to a PC using a USB-serial cable, or (if it has an RS232 port) a level-converter circuit - see above for details. When this is done, you will need to set up a terminal emulator (http://en.wikipedia.org/wiki/Terminal_emulator) program on your PC as described below.

Console serial parameters

The following parameters are needed to connect to the Pi console, and apply on both Linux and Windows.

- Speed (baud rate): 115200
- Bits: 8
- Parity: None
- Stop Bits: 1
- Flow Control: None

Linux terminal set up

If your PC is running Linux, you will need to know the *port name* of its serial port:

- Built-in (standard) Serial Port: the Linux standard is **/dev/ttyS0**, **/dev/ttyS1**, and so on
- USB Serial Port Adapter: **/dev/ttyUSB0**, **/dev/ttyUSB1**, and so on.
    - Some types of USB serial adapter may appear as **/dev/ttyACM0** ...

You will need to be a member of the *dialout* group to access this port (for later releases the required group is *tty*). You can check which is needed with:

```
ls -l /dev/ttyUSB0
```

and you will see something like "crw-rw----T 1 root dialout ...", *c* means character device, and root can 'read,write' and the group *dialout* can 'read,write' to the port and everyone else cannot access it.

To find out if you, the current user, is in the group dialout, use the command:

```
id
```

If you do not see *dialout* listed, add yourself with the command

```
sudo usermod -a -G dialout username
```

You then have a choice of terminal emulation programs:

- **Super Easy Way Using GNU Screen**

Enter the command below into a terminal window

```
screen port_name 115200
```

To exit GNU screen, type Control-A k.

- **Super Easy Way Using Minicom**

Run minicom with the following parameters:

```
minicom -b 115200 -o -D Port_Name
```

You can exit minicom with Control-A x

- **Tedious Old-Fashioned Way Using Minicom**

Another method to setup *minicom* is described in the Tincantools Minicom Tutorial (http://www.tincantools.com/wiki/Minicom)

- **GUI method with GtkTerm**

Start *GtkTerm*, select Configuration->Port and enter the values above in the labeled fields.

Network connection with the point-to-point protocol (ppp)

The easiest way to set up a network connection between your Raspberry Pi and another computer is with an ethernet cable. If this is not possible, as is the case for the Raspberry Pi Model A, you can set up a connection over the serial cable. This uses the Point-to-point Protocol (PPP). A network connection running over a serial cable can be very useful for copying files onto the Raspberry Pi.

Step 1: login to the Raspberry Pi over the serial cable and run the Point-to-Point Protocol Daemon:

```
sudo pppd noauth
```

Some garbage will start appearing in the terminal. This is the cue to quit your terminal program and proceed to step two.

Step 2: on your local computer, start the Point-to-Point protocol. On a Linux or Mac computer you can do this by typing:

```
sudo pppd noauth proxyarp /dev/tty.usbserial-FTGCC2MV 115200 10.0.0.1:10.0.0.2 passive local maxfail 0 nocrtscts xonxoff
```

replacing /dev/tty.usbserial-FTGCC2MV with the name of your serial port. In the above line, 115200 is the baud rate of the connection, 10.0.0.1 is the local internet protocol (IP) address, the address you want your computer to have. 10.0.0.2 is the remote IP address, it is the address that the Raspberry Pi will have.

Test the connection:

```
ping 10.0.0.2
```

Virtual connection to the LAN

Instead of 10.0.0.0/8 you could as well use normal 192.168.0.0/16 addresses; the first address must be the real address of the local (serving) system. You can chose the second address; it must not yet be assigned on the LAN (and be outside the DHCP range). The advantage is that the system connected to the serial line will appear as if it is directly connected to the LAN (arp protocol).

You must enable routing on the system directly connected to the LAN for other systems to access the system connected to the serial line:

```
sudo sysctl -w net.ipv4.ip_forward=1
```

On the guest system connected via the serial cable you must set the default route pointing to the serving system, e.g.

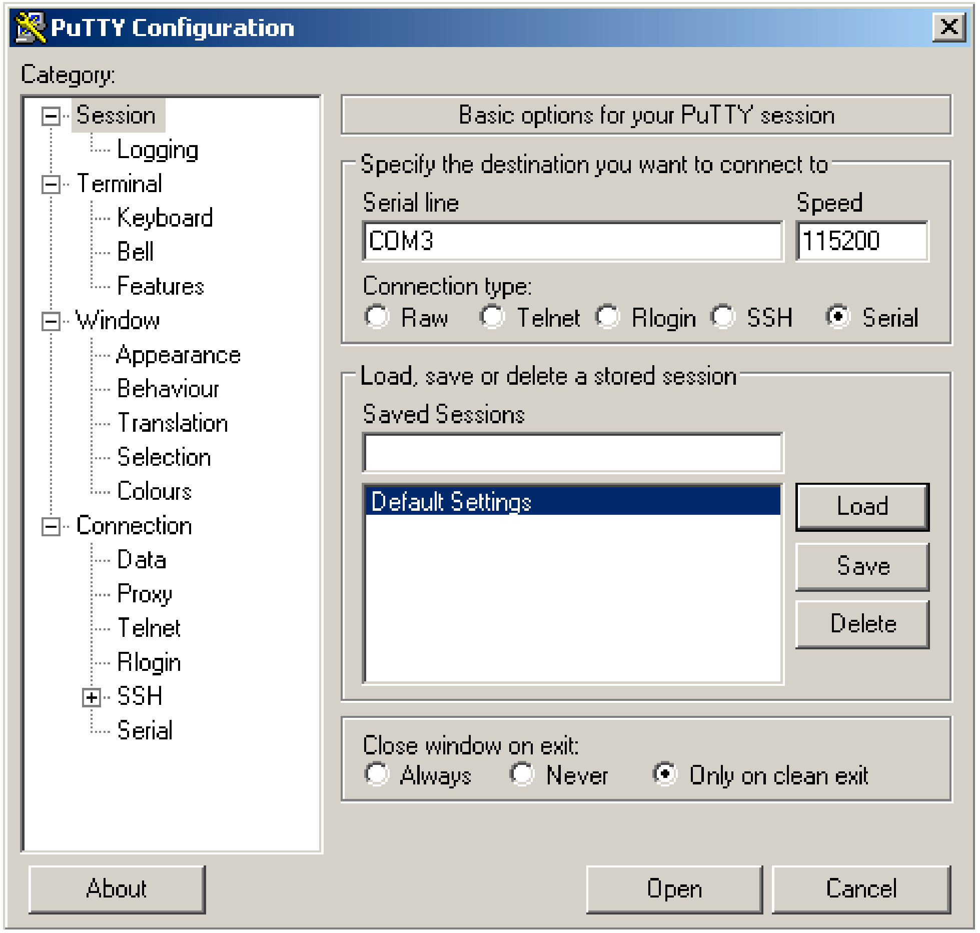
```
sudo route add default gateway 192.168.1.21
```

You should also configure /etc/resolv.conf if you want to use DNS.

Windows terminal set-up

Users of Windows Vista or later will need to download a terminal program, for instance PuTTY (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html), or TeraTerm (http://en.wikipedia.org/wiki/Tera_Term). Users of XP and below can choose between using *PuTTY* and the built-in *Hyperterminal.*

PuTTY users simply need to choose 'serial', select the correct COM port and set the speed, as shown in the dialog below.



Specify the destination you want to connect to

Serial line: COM3

Speed: 115200

Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Load, save or delete a stored session

Saved Sessions

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

If you are unsure of the COM port, run [Device Manager (https://en.wikipedia.org/wiki/Device_manager)] and look under 'Ports'. USB-attached serial adapters should have the name of the adapter shown (the Adafruit cable comes up as 'Prolific USB-to_Serial Comm Port'.

Boot messages

If your connection is set up correctly, when the Pi is booted you should see many messages as the system comes up:

```
Uncompressing Linux... done, booting the kernel.
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Linux version 3.2.27+ (dc4@dc4-arm-01) (gcc version 4.7.2 20120731 (prerelease) (crosstool-NG linaro-1.13.1+bzr2458 - Linaro GCC 2012
[    0.000000] CPU: ARMv6-compatible processor [410fb767] revision 7 (ARMv7), cr=00c5387d
[    0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT nonaliasing instruction cache
[    0.000000] Machine: BCM2708
[    0.000000] Memory policy: ECC disabled, Data cache writeback
[    0.000000] Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 113792
[    0.000000] Kernel command line: dma.dmachans=0x7f35 bcm2708_fb.fbwidth=656 bcm2708_fb.fbheight=416 bcm2708.boardrev=0xf bcm2708.serial=0xcc5c4b6
```

and so on. Eventually, you should see a login prompt:

```
Debian GNU/Linux wheezy/sid raspi2 ttyAMA0

raspi2 login:
```

You can then log in as you would with a keyboard and screen.

Unwanted serial garbage input

Note that on older software by accident the internal pullups of the RxD GPIO pins were not enabled, this could lead to lots of serial garbage being picked up if the GPIO pin was touched, or even if a finger was nearby. In extreme case this could lead to kernel warnings and other problems.

Connection to a microcontroller or other peripheral

The TxD and RxD signals can also be connected directly to similar signals on a microcontroller board like the Arduino provided the signals are all at 3V3 levels. Some Arduino variants use 5V levels, and should have level conversion (see 'Connections and Signal levels' above). Even at 3V3 levels, it's still a good idea to put 2K2 series resistors in the lines to prevent damage when two outputs are connected together, which could also happen if a GPIO input pin is accidentally programmed as output.

If your microcontroller uses 5V logic levels, level conversion is usually necessary - see 'Connecting to a PC' for details.

Preventing Linux using the serial port

The Broadcom UART appears as `/dev/ttyAMA0` under Linux. There are several minor things in the way if you want to have dedicated control of the serial port on a Raspberry Pi.

UPDATE: There's now a nice little script to automate all the steps below, making them unnecessary. Read more about it here (https://github.com/lurch/rpi-serial-console)

- Firstly, the kernel will use the port as controlled by kernel command line contained in `/boot/cmdline.txt`. The file will look something like this:

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

The *console* keyword outputs messages during boot, and the *kgdboc* keyword enables kernel debugging. You will need to remove all references to ttyAMA0. So, for the example above `/boot/cmdline.txt`, should contain:

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

You must be root to edit this (e.g. use `sudo nano /boot/cmdline.txt`). Be careful doing this, as a faulty command line can prevent the system booting.

- Secondly, after booting, a login prompt appears on the serial port. This is controlled by the following lines in `/etc/inittab`:

```
#Spawn a getty on Raspberry Pi serial line
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

You will need to edit this file to comment out the second line, i.e.

```
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Finally you will need to reboot the Pi for the new settings to take effect. Once this is done, you can use `/dev/ttyAMA0` like any normal linux serial port, and you wont get any unwanted traffic confusing the attached devices.

The above instructions have been verified on Raspbian 'wheezy'; other distributions may be set up differently. To double-check, use

```
cat /proc/cmdline
```

to show the current kernel command line, and

```
ps aux | grep ttyAMA0
```

to search for `getty` processes using the serial port.

A tutorial on accessing the Pi's serial port from Python is available at Serial_port_programming.

Handshaking lines

You can have the RTS0 signal on GPIO 17 (P1-11) or GPIO 31 (P5-06) if you set them to ALT function 3. Likewise, the CTS0 is available on GPIO 30 (P5-05), if it is set to ALT function 3. You can control the settings of IOs with gpio_setfunc (https://github.com/rewolff/bw_rpi_tools/tree/master/gpio)

Glitch when opening serial port

When the serial port is opened the voltage on TXD pulses negative for approximately 32 us (regardless of the baud rate). This pulse may be interpreted as a transmission by a device connected to the TXD pin, which could have unintended effects. An error tolerant communication protocol should be used to avoid problems this glitch could cause. Another method for avoiding problems is to use a GPIO pin to implement the RTS signal, and to have the connected device ignore all data on TXD until RTS is asserted. If the connected device is susceptible to the glitch and cannot be modified, it is sometimes possible to obtain correct operation by opening the serial port in advance of initiating transmission. This can be done in the shell with the sleep program:

```
sleep infinity >/dev/ttyAMA0 &
```

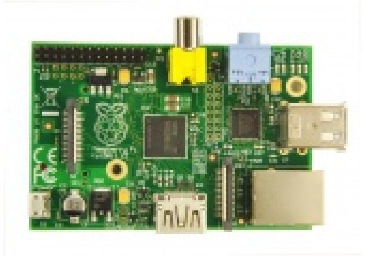
In a shell script, the following commands may be used to kill the sleep process once serial transmission is complete.

```
sleep infinity >/dev/ttyAMA0 &
sleep_pid=$!
disown $sleep_pid # this is required to prevent an error message when sleep is terminated
# Use serial port here
kill $sleep_pid
```

Raspberry Pi

- V
- T
- E (http://elinux.org/index.php?title=Template:Raspberry_Pi&action=edit)

Startup	Buying Guide - SD Card Setup - Basic Setup - Advanced Setup - Beginners Guide - Troubleshooting
Hardware	Hardware - Hardware History - Low-level peripherals - Expansion Boards
Peripherals	Screens - Cases - Other Peripherals (Keyboard, mouse, hub, wifi...)
Software	Software - Distributions - Kernel - Performance - Programming - VideoCore APIs - Utilities
Projects	Tutorials - Guides - Projects - Tasks - DataSheets - Education - Communities



Retrieved from "http://elinux.org/index.php?title=RPi_Serial_Connection&oldid=343256"
Category: RaspberryPi