Accessing an RS-232 serial interface from RaspberryPi

Luis Kornblueh

July 20, 2015

1 Hardware required

In addition to the audio, video, network and USB connectors, the Raspberry Pi also has the GPIO pins. These pins include an UART serial console, which can be used to log in to the Pi, and many other things. However, normal UART device communicate with -12V (logical 1) and +12V (logical 0), which may destroy the RaspberryPi, which operates at 3.3V. Even the normal TTL level serial connection works at 5V and may damage the RaspberryPI.

To overcome this problem a MAX3232CPE transceiver can be used to safely convert the normal UART voltage levels to 3.3V for the Raspberry Pi. What we do need is:

- a MAX3232CPE RS-232 to 3.3V logic level transceiver (or similar IC)
- five 0.1 μF capacitors (following the MAX3232CPU datasheet ceramic are fine)
- jumper wires and a small breadboard
- a 9 pin SUB-D plug

The connections on the RaspberryPi side are straightforward. Use the 3.3V pin for power (the current should not exceed 50 mA, should not be an issue since the MAX3232CPE draws less than 1 mA and the capacitors are small) and GND and the two UART pins, TXD and RXD.

The MAX3232CPE uses a few capacitors to deliver true +-12V RS-232 signalling on one end and 3.3V on the other. Be referred to the datasheet of this chip.

Above you can see one rather compact way to wire the MAX3232. The orange, white, green and black wires come from Raspberry Pi and provide power and data lines. The red, brown and blue wires go to the RS-232 port. See the illustration on right for connections on this side.

The RS-232 connection diagram is from the side you would solder the wires from back side of the connector and wired so you can connect a PC USB serial adapter to the Raspberry Pi. If you would like to talk to serial peripherals from the RaspberryPi instead, RX/TX wires need to be reversed.

Before you connect the Pi, check with a voltmeter that GND from Raspberry Pi and GND from RS-232 do not differ from each other too much (a few millivolts is usually OK), otherwise you may risk a ground loop and damage to your equipment!

1.1 Starting Linux Screen

Screen is started from the command line just like any other command:

```
[root@office ~]# screen
```

You are now inside of a window within screen. This functions just like a normal shell except for a few special characters.

1.2 Control Command

Command: Ctrl-a

Screen uses the command Ctrl-a that is the control key and a lowercase a as a signal to send commands to screen instead of the shell.

For example, Ctrl-a then ?. You should now have the screen help page.

```
Screen key bindings, page 1 of 4.

Command key:  ^A   Literal ^A:  a

break       ^B b          only        Q
clear       C             other       ^A
colon       :             pow_break   B
copy        ^[ [          pow_detach  D
detach      ^D d          prev        ^P p ^?
digraph     ^V            readbuf     <
displays    *             redisplay   ^L l
fit         F             removebuf   =
flow        ^F f          reset       Z
focus       ^I            screen      ^C c
hardcopy    h             select      '
help        ?             silence     _
```

Key bindings are the commands the screen accepts after you hit Ctrl-a. You can reconfigure these keys to your liking using a .screenrc file, but I just use the defaults.

1.3 Creating Windows

Command: Ctrl-a c.

To create a new window, you just use Ctrl-a c.

This will create a new window for you with your default prompt. Your old window is still active.

For example, I can be running top and then open a new window to do other things. Top stays running! It is still there. To try this for yourself, start up screen and then run top. (Note: I have truncated some screens to save space.)

Start top

```
top - 09:10:33 up 35 days, 17:26,  1 user,  load averag
Tasks: 131 total,   1 running, 130 sleeping,   0 stoppe
Cpu(s):  0.4%us,  0.2%sy,  0.0%ni, 99.4%id,  0.0%wa,  0
Mem:  12302040k total,  6363652k used,  5938388k free,
Swap:  1052248k total,       12k used,  1052236k free,
```

Now open a new window with: Ctrl-a c

Your top window is still running you just have to switch back to it.

1.4 Switching Between Windows

Command: Ctrl-a n

Screen allows you to move forward and back. In the example above, you could use Ctrl-a n to get back to top. This command switches you to the next window.

The windows work like a carousel and will loop back around to your first window.

You can create several windows and toggle through them with Ctrl-a n for the next window or Ctrl-a p for the previous window.

Each process will keep running until you kill that window.

1.5 Detaching From Screen

Command: Ctrl-a d

Detaching is the most powerful part of screen. Screen allows you to detach from a window and reattach later. If your network connection fails, screen will automatically detach your session!

You can detach from the window using Ctrl-a d.

This will drop you into your shell.

All screen windows are still there and you can re-attach to them later.

This is great when you are using rsync for server migration.

1.6 Reattach to Screen

If your connection drops or you have detached from a screen, you can re-attach by just running: 1

```
[jeffh@office ~]$ screen -r
```

This will re-attach to your screen.

However, if you have multiple screens you may get this:

```
[jeffh@office ~]$ screen -r
```

```
There are several suitable screens on:
31917.pts-5.office      (Detached)
31844.pts-0.office      (Detached)
Type "screen [-d] -r [pid.]tty.host" to resume one of them.
```

If you get this, just specify the screen you want.

```
[jeffh@office ~]$ screen -r  31844.pts-0.office
```

1.7 Logging Your Screen Output

As a consultant, I find it important to keep track of what I do to someone's server. Fortunately, screen makes this easy.

Using Ctrl-a H, creates a running log of the session.

Screen will keep appending data to the file through multiple sessions. Using the log function is very useful for capturing what you have done, especially if you are making a lot of changes. If something goes awry, you can look back through your logs.

1.8 Getting Alerts

Screen can monitor a window for activity or inactivity. This is great if you are downloading large files, compiling, or waiting for output.

If you are waiting for output from a long running program, you can use Ctrl-a M to look for activity. Screen will then flash an alert at the bottom of the page when output is registered on that screen.

I use this when running a command that takes a long time to return data. I can just fire up the command, switch to another window and not have to keep switching back to check the status.

You can also monitor for inactivity. Why use this?

If you are downloading a large file or compiling a program, you can be notified when there is no more output. This is a great signal to when that job is done. To monitor for silence or no output use Ctrl-A _.

1.9 Locking Your Screen Session

If you need to step away from your computer for a minute, you can lock your screen session using Ctrl-a x. This will require a password to access the session again.

```
Screen used by root <jeffh>.
Password:
```

1.10 Stopping Screen

When you are done with your work, I recommend you stop the session instead of saving it for later. To stop screen you can usually just type exit from your shell. This will close that screen window. You have to close all screen windows to terminate the session.

You should get a message about screen being terminated once you close all windows.

```
[screen is terminating]
```

Alternatively, you can use Ctrl-a k. You should get a message if you want to kill the screen.