

How to Use a Pro Micro as a Cheap Soarer's Converter

Soarer's Converter has traditionally used a Teensy micro-controller board but these are very expensive, upwards of fifteen pounds. It is however possible to use a sub-£3 Pro Micro clone off e-Bay or AliExpress. These have fewer pins but still enough as a Soarer's Converter only NEEDS four; the rest are for added functions like auxiliary buttons and LEDs which you may not need or want.

A patched firmware is available from <https://deskthority.net/resources/slightly-changed-sc-v1-12/15676> which makes the Pro Micro's PB6 pin work as the Reset line. Most keyboards don't use this but we may as well have the option. Using this, the only thing we'll be missing from the Teensy version is Auxiliary Button 1.

Disclaimer

This guide tells you how to set-up the board on a Linux system. I'm not getting into the silly arguments about whether Linux or Windows is 'better' as each is better for some things. Linux wins hands-down when it comes to working with micro-controllers though. After the converter is initially programmed it can be used on any system and any additional configuration is actually easier to do on Windows.

Get a Pro Micro clone

This part is easy. You can get them on eBay for about £3 shipped, and also on AliExpress at a slightly lower price. Look at the small print of the listing though: searches can bring up some cheaper ones with the wrong chip and some that run at 8MHz on a 3.3V supply. We want one with an ATmega32u4 chip than runs at 16Mhz from a 5V supply.

Flash the Firmware

Download the firmware above and extract the file [Soarer_at2usb_v1.12_atmega32u4_ProMicro_Reset.hex](#) to your home directory.

In order to flash the device, you need to figure out which "serial port" your device is attached to. First of all, before connecting the Pro Micro, identify what serial devices (including virtual ones) are currently connected to your machine with this command:—

```
ls /dev/tty*
```

On my machine, this produces the following:—

```
joe@joe-desktop ~ $ ls /dev/tty*
/dev/tty      /dev/tty23  /dev/tty39  /dev/tty54  /dev/ttyS10 /dev/ttyS26
/dev/tty0     /dev/tty24  /dev/tty4   /dev/tty55  /dev/ttyS11 /dev/ttyS27
/dev/tty1     /dev/tty25  /dev/tty40  /dev/tty56  /dev/ttyS12 /dev/ttyS28
/dev/tty10    /dev/tty26  /dev/tty41  /dev/tty57  /dev/ttyS13 /dev/ttyS29
/dev/tty11    /dev/tty27  /dev/tty42  /dev/tty58  /dev/ttyS14 /dev/ttyS3
/dev/tty12    /dev/tty28  /dev/tty43  /dev/tty59  /dev/ttyS15 /dev/ttyS30
/dev/tty13    /dev/tty29  /dev/tty44  /dev/tty6   /dev/ttyS16 /dev/ttyS31
/dev/tty14    /dev/tty3   /dev/tty45  /dev/tty60  /dev/ttyS17 /dev/ttyS4
/dev/tty15    /dev/tty30  /dev/tty46  /dev/tty61  /dev/ttyS18 /dev/ttyS5
/dev/tty16    /dev/tty31  /dev/tty47  /dev/tty62  /dev/ttyS19 /dev/ttyS6
/dev/tty17    /dev/tty32  /dev/tty48  /dev/tty63  /dev/ttyS2  /dev/ttyS7
/dev/tty18    /dev/tty33  /dev/tty49  /dev/tty7   /dev/ttyS20 /dev/ttyS8
/dev/tty19    /dev/tty34  /dev/tty5   /dev/tty8   /dev/ttyS21 /dev/ttyS9
/dev/tty2     /dev/tty35  /dev/tty50  /dev/tty9   /dev/ttyS22
/dev/tty20    /dev/tty36  /dev/tty51  /dev/ttyprintk /dev/ttyS23
/dev/tty21    /dev/tty37  /dev/tty52  /dev/ttyS0  /dev/ttyS24
/dev/tty22    /dev/tty38  /dev/tty53  /dev/ttyS1  /dev/ttyS25
joe@joe-desktop ~ $
```

Take a note of the output, we'll need it later.

If You've Never Flashed Your Device Before

If this is your first time flashing the Pro Micro, this part will be easy. On the Pro Micro's first boot, instead of trying to run code (that you haven't uploaded yet!), it just sticks in bootloader mode. So if you only want to flash firmware once and forget about it – good news.

Just plug in your device and it should automatically go into bootloader mode.

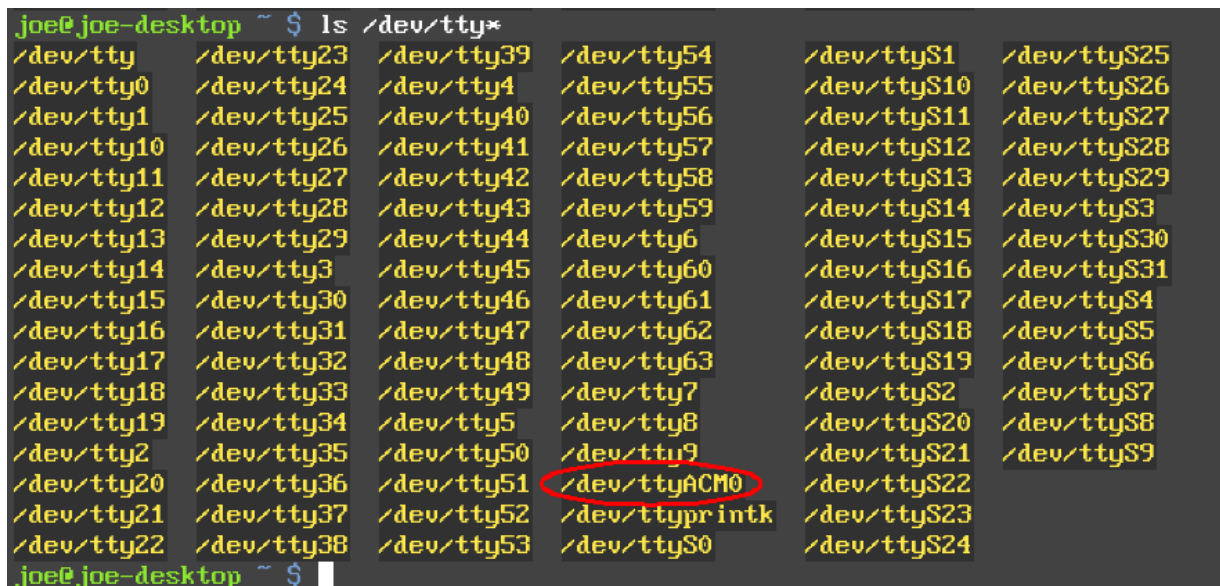
If You Have Previously Flashed Your Device

If you've previously flashed your Pro Micro, instead of going straight into bootloader mode it will try to run the code that you uploaded to it. To go into bootloader mode, you need to quickly connect the GND + RST pins twice, just as the device powers on.

Next, re-run the command, the same as before:–

```
ls /dev/tty*
```

There should be one more output device than was seen previously. For me, it's `/dev/ttyACM0`.



```
joe@joe-desktop ~ $ ls /dev/tty*
/dev/tty      /dev/tty23  /dev/tty39  /dev/tty54      /dev/ttyS1    /dev/ttyS25
/dev/tty0     /dev/tty24  /dev/tty4   /dev/tty55      /dev/ttyS10   /dev/ttyS26
/dev/tty1     /dev/tty25  /dev/tty40  /dev/tty56      /dev/ttyS11   /dev/ttyS27
/dev/tty10    /dev/tty26  /dev/tty41  /dev/tty57      /dev/ttyS12   /dev/ttyS28
/dev/tty11    /dev/tty27  /dev/tty42  /dev/tty58      /dev/ttyS13   /dev/ttyS29
/dev/tty12    /dev/tty28  /dev/tty43  /dev/tty59      /dev/ttyS14   /dev/ttyS3
/dev/tty13    /dev/tty29  /dev/tty44  /dev/tty6       /dev/ttyS15   /dev/ttyS30
/dev/tty14    /dev/tty3   /dev/tty45  /dev/tty60      /dev/ttyS16   /dev/ttyS31
/dev/tty15    /dev/tty30  /dev/tty46  /dev/tty61      /dev/ttyS17   /dev/ttyS4
/dev/tty16    /dev/tty31  /dev/tty47  /dev/tty62      /dev/ttyS18   /dev/ttyS5
/dev/tty17    /dev/tty32  /dev/tty48  /dev/tty63      /dev/ttyS19   /dev/ttyS6
/dev/tty18    /dev/tty33  /dev/tty49  /dev/tty7       /dev/ttyS2    /dev/ttyS7
/dev/tty19    /dev/tty34  /dev/tty5   /dev/tty8       /dev/ttyS20   /dev/ttyS8
/dev/tty2     /dev/tty35  /dev/tty50  /dev/tty9       /dev/ttyS21   /dev/ttyS9
/dev/tty20    /dev/tty36  /dev/tty51  /dev/ttyACM0    /dev/ttyS22
/dev/tty21    /dev/tty37  /dev/tty52  /dev/ttyprintk  /dev/ttyS23
/dev/tty22    /dev/tty38  /dev/tty53  /dev/ttyS0      /dev/ttyS24
joe@joe-desktop ~ $
```

To flash the device, you need to have AVRdude installed. You can do this with your normal package manager or with the command `sudo apt-get install avrdude`.

Once it's installed, confirm that it works by just running `avrdude` on the command-line. It should throw some errors. If so, it works.

Once you have AVRdude set up, navigate to the directory with your `.hex` file in it. Then, run the following as one long line remembering that both Linux and AVRdude are case-sensitive:–

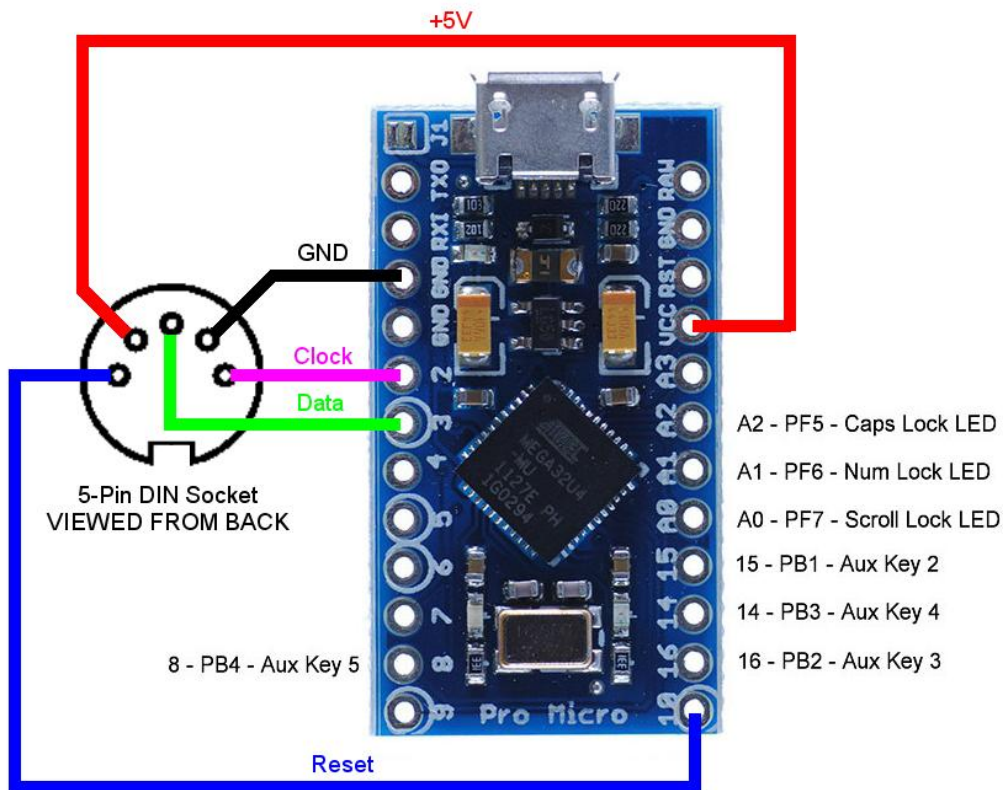
```
avrdude -p atmega32u4 -P YOUR_SERIAL_PORT -c avr109 -U
flash:w:Soarer_at2usb_v1.12_atmega32u4_ProMicro_Reset.hex
```

Of course, replace `YOUR_SERIAL_PORT` with your serial port's device name as discovered earlier, eg `/dev/ttyACM0`.

If it says "device not in sync" or similar, your device is no longer in bootloader mode. Unplug it, and get it back into bootloader mode like you did in the previous step, and try again.

I already had AVRdude installed as part of the Arduino IDE and found it wouldn't work. Eventually I solved this by renaming the file `.avrduderc` in my home directory. It doesn't matter what it's renamed to and after finishing this job put it back to what it was. If you do this via a graphical file manager you may have to press `Ctrl-h` to make it show up.

Wiring



Plug the keyboard into the converter before plugging the converter into the computer.

The keyboard and converter should now be basically working. To further configure it – re-map keys, set-up macros &c – download Soarer's original file-set from <https://deskthority.net/resources/file/6142> which contains instructions and the tools to do so.

More comprehensive instructions may be found at:–

<https://www.dropbox.com/s/ljooth6e5hhzyh2/Soarer%27s%20Converter%20-%20My%20Simple%20Terms%20-%20Work%20in%20Progress.txt?dl=0>