SimpleGpUsb

06/06/2009

SimpleGpUsb is my home-brew auto guider interface, which I decided to create when I saw how much the commercial guiding products cost. I'm sure that they are fine pieces of equipment, but I couldn't help but think that they were over priced, and that I could probably make the same thing for considerably less.

This document is going to summarize my project, and hopefully provide you with all the information you need to make your own SimpleGpUsb should you want to. Rather than reproduce all of the reference material I used during this project I shall hyperlink in any relevant information to their original sources.

I place all of my ideas and source code into the public domain under the GPL v3 license, so feel free to tweak, modify and enhance to suit your own personal needs. Just make sure that you share your new creations with the world and give credit where credit is due. A copy of the GPL license is included with the source code.

Hand Controller Modification

Figure 1: EQ3-2 Hand Controller Modification

First things first... To be able to connect the SimpleGpUsb to my EQ3-2 mount I needed to perform the same Hand Controller Modification as detailed for the Shoestring products. Their instructions are excellent, and although it was a bit scary taking my precious (and expensive) controller apart, it was all fairly straight forward.

Rather than buy the ShoeString kit I made up my own connector so that I had a proper RJ-12 socket coming out of the controller instead of a plug lead that required a coupler. I just got 5 lengths of suitably coloured wire, an RJ-12 socket, some heat shrink and a little plastic box to house it in. It's almost exactly the same as the ShoeString approach, I just preferred doing it this way.

The socket is wired as shown in Figure 2. It's obviously very important that you get your wiring correct or the whole thing will fail to work, and potentially damage your hardware!

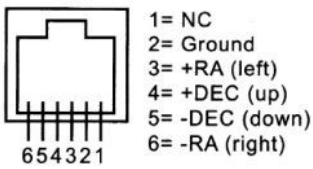

Figure 2: RJ-12 Socket Pins

I have done my best to follow the ST-4 “standard” but it seems that there is a fair amount of variance in the way it is actually implemented. Ultimately you don't need to adhere to the standard if you don't want to, so long as all the connections all tie up i.e. the ground wire on the hand controller goes to the ground wire on the SimpleGpUsb etc.

Note: I actually left this step till last when I did it, as I wanted to make sure that my device actually worked first! You might want to do the same, I wanted to point out this necessary step up front in case it puts the fear of God into you!

PC Hardware Interface

My first requirement was a USB interface board. USB is all my laptop has, so serial and parallel options were out from the start. After a quick Google I found this little gem, the CP2103 breakout board. This small, simple, and best of all cheap, device is just perfect for this project. The 4 general purpose input/output pins will control the 4 directions of movement (RA+, DEC+, DEC-, RA-).

Figure 3: The CP2103 Breakout Board

You'll need configure the device by running a utility provided by the company that designs and manufactures the board, Silicon Labs. For your convenience their utilities are included in the files download that accompanies this document. They can be found in the **Silicon Labs** folder.

You will find an application called **CP210xPortConfig.exe**. Just run the utility, with the CP2103 board attached to the computer, and make sure that the settings are the same as in Figure 4.

This step is very important! The key item here is to ensure that the **GPIO Reset Mode** is configured as **Push-Pull**. This little setting caused me about a week of tearing my hair out trying to figure out why a seemingly simple circuit just wouldn't work!

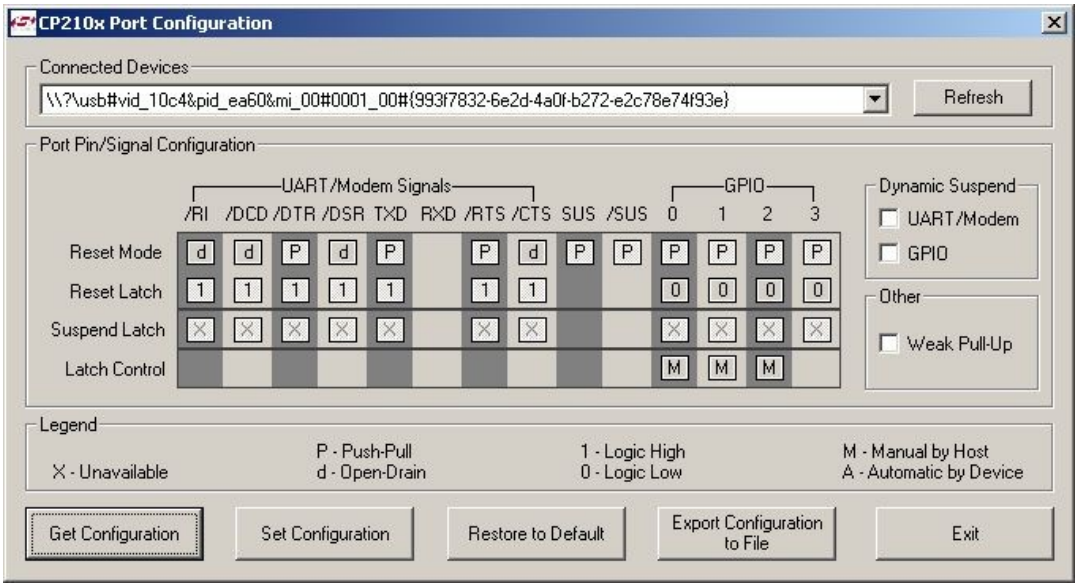

Figure 4: The Port Configuration Utility

Another customization feature that you can add is to set the **Product String** within the EEPROM of the CP2103 board. It's not an essential step, but it makes the whole thing look a little more professional.

This cosmetic change is made by running the **CP210xSetIDs.exe** utility, with the CP2103 board attached to the computer, to program the device with a custom **Product String** i.e. SimpleGpUsb.

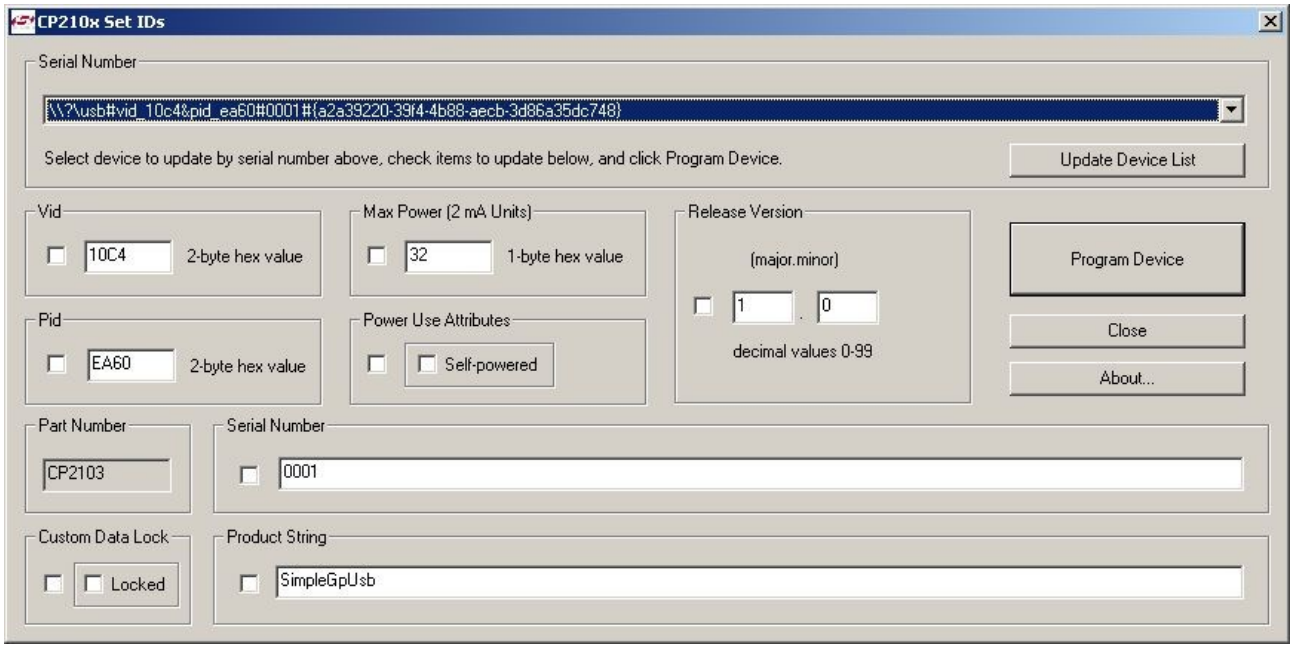

Figure 5: The Set Ids Utility

I left all of the other settings at their defaults.

The Circuit

This circuit used is very simple, and is built around a Darlington Array. This is basically a convenient packaging of 7 transistors into one chip. I used four of these transistors as switches to make connections that simulate button presses on the hand controller.

Conceptually it's important to note that the SimpleGpUsb does not output any current, it's just a switching system to make the “loop” within the hand controllers own circuit, which was tapped into when making the hand controller mod.

Figure 6 shows the circuit design, with the Darlington Array in the middle of the action. The general purpose input output (GPIO) pins from the CP2103 breakout board act as the input to the top 4 transistors in the array, on the left side of the chip. I arbitrarily decided that the following functions would be assigned to the GPIO pins, as it roughly follows the order of the RJ-12 sockets pins:

- GPIO pin 0 = Right Ascension +
- GPIO pin 1 = Declination +
- GPIO pin 2 = Declination -
- GPIO pin 3 = Right Ascension -

On the opposite side of the chip the output pins should be connected to the appropriate pins of the RJ-12 socket as detailed in Figure 2 above. The ground wire from the hand controller circuit should be connected to pin 8 at the bottom left of the array.

With these connections in place, any GPIO pins toggled to an “on state” will trigger the transistor switch and thus allow the current to flow through hand controller circuit, just as if a button had been physically pressed.

The Darlington Array I used was a **ULN2003A** that I picked up from my local Maplin.

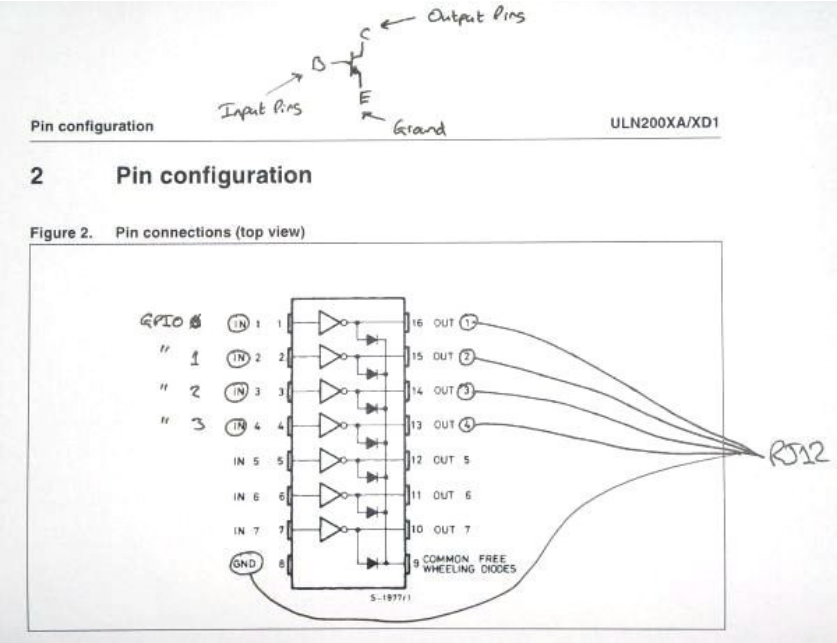

Figure 6: The Basic Circuit

Putting It Together

To make the final product I used a small plastic enclosure from Maplin. It measures approximately 75 x 55 x 25mm and is just perfect for all the odds and ends to fit in. I had to do a bit of cutting to make holes for the USB input socket, and the RJ-12 socket, but nothing too drastic. I affixed the CP2103 breakout board in place with some self adhesive plastic mounting pins that I picked up from an electronic components shop.

As you can see it really is quite simple in the end. The 4 GPIO pins go to the 4 inputs on the Darlington Array, and then the wires from the RJ-12 socket go to the corresponding output pins, and of course to ground. And that's pretty much it!

Figure 7: Internal Layout

Cables

In trying to stick with ST-4 de facto standard for guiding products, I needed to make sure that I had the right sort of RJ-12 cable to connect the modified hand controller to the SimpleGpUsb.

Another quick Google brought up some nice documents, such as this one from ShoeString: Guide Port Cabling. The cable and sockets used are of the RJ-12 variety, which can also be referred to as **6P6C** (6 positions, 6 connectors).

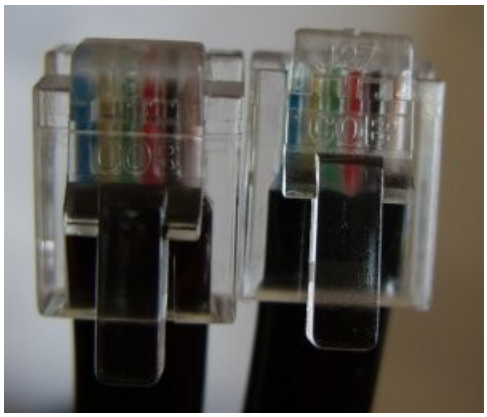

Figure 8: Guiding Cable

With the sockets wired as shown in Figure 2, I needed a cable wired like the one shown in Figure 8.

The crucially important thing to note here is the fact that the wires are in the same order from left to right on both plugs. This means that the ground pin on one socket will connect to the ground pin on the other socket.

Software Support

For the SimpleGpUsb to be of any use I had to make sure that the freely available guiding software such as PHD, and GuideDog could “talk” to it. Fortunately the ASCOM Initiative came to my rescue here, with their industry standard for all things astronomical.

By implementing the “Telescope” interface defined by them, software such as PHD and GuideDog can easily communicate with the SimpleGpUsb, without having to know anything about it. This should hold true for any other ASCOM compliant guiding software that you might use.

Before you can use my driver you must ensure that the ASCOM Platform Version 5a is installed on your machine. Then you can install the driver I developed for the SimpleGpUsb, which is included in the files download that goes with this document. The installer should configure everything you need to get your computer to communicate with your hardware once it's all built and ready to go.

The driver is very simple in nature, and only supports a small subset of the available features, just what is needed for guiding. There are a few configuration options that you can adjust via the properties dialog:

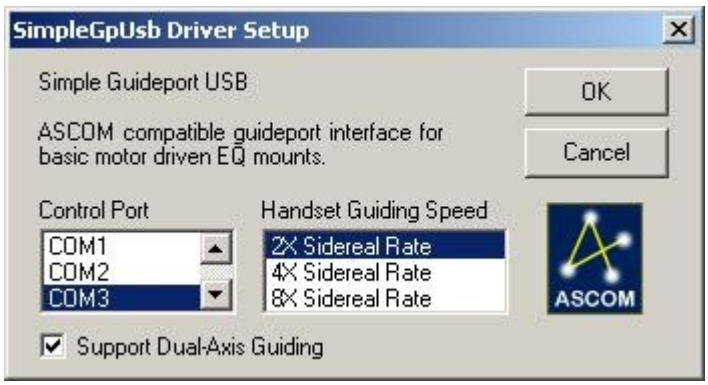

Figure 9: ASCOM Driver Properties

First and most important of all is the COM port that the SimpleGpUsb declares itself as being. You might need to go into the windows device manager and look under the ports item to find this out.

Next you can specify the guiding speed. On the hand controller for my EQ3-2 there is a switch that offers 2x, 4x and 8x sidereal rate, and this option reflects those values. It is very important that the switch position on the hand controller and this software option be the same, otherwise all of the guiding corrections will be wrong and you'll find the scope chasing around in circles! Generally you will want to use the 2x rate to make smaller and more refined adjustments. See Appendix A for notes about the guiding rates.

Finally you can specify if the declination axis is supported for guiding with the “Support Dual-Axis Guiding” option. If your mount only has a Right Ascension motor you should not tick this.

The Finished Product

Here are a few pictures, one of the finished device, one of my cheap and cheerful guiding set up, and finally the first ever guided image I captured!

Figure 10: The Finished SimpleGpUsb

Figure 11: My Modest Guiding Set up

Figure 12: My First Guided Image (5 Minute Exposure)

Disclaimer

I hope that my instructions in this document have been clear, and I have done my best to not make any mistakes. However I am only human, so this cannot be ruled out. If you choose to copy what I have done I accept no responsibility for any damages or loss you may suffer. I offer this information freely to anyone who might like to use it, but I offer no warranty or promise of its fitness.

It all worked like a dream for me, and I hope that it does for you too, but I've got to cover my back! ;)

Appendix A

Sidereal rate is ¼ of a degree per minute, or 15 arcseconds. With this knowledge we can see that the guiding rates of the EQ3-2 hand controller of 2x, 4x and 8x sidereal rate equate to the following:

- 2x = ½ a degree per minute
- 4x = 1 degree per minute
- 8x = 2 degrees per minute

I have discovered one slight issue with these rates with regard to my EQ3-2 motor drives and controller. Basically these speeds are for the “forward” motor direction, but when in reverse the actual speeds are 1x sidereal slower than the above values. So the reverse rates are 1x (the motor stops so that the Earth's rotation takes effect), 3x and 7x. This difference is ¼ of a degree per minute.

Unfortunately this slight discrepancy cannot be accounted for in the ASCOM driver as it is assumed that the forward and reverse speeds will be identical for a given rate.

In practice I have discerned no ill effects of this small variance, but it is there and should be considered.