ServoMaster: Observations on Pololu Servo Controllers

Table of contents

1. Summary

All Pololu servo controllers share elegant and compact mechanical design with lots of status lights - should be pretty comfortable to work with.

All of them share the same protocol, so the driver doesn't have to be changed. The only difference is the number of servos they support - 8 or 16.

Also, all of them support the Mini SSC II compatibility mode - this helps a lot if you have some SSCs laying around and want a drop-in upgrade.

There are two classes of devices - pure serial and dual interface. The only one I have in my possession at the moment is the dual interface one, but one of my contacts is working with the serial 16-Servo controller, so I believe reasonably accurate conclusions may be drawn.

Individual Servo min/max preset
Silent Operation
Hardware controlled constant velocity transition, per servo

Table 1: Supported Features

Warning:
Not all the features are implemented yet.

2. Common Features

2.1. Good Things

- Elegant mechanical design - all 16 servo connectors are packed nicely, the USB connector is a mini connector (unlike full B connector found on other devices);
- Small - much smaller than other designs;
- Lots of diagnostic lights on the PCB;
- Precise. The resolution is 5000 steps, which should probably give you more control over your mechanical components than they'd notice;
- Fast. The baud rate may go as high as 38.4K.
- Flexible. You can control properties (shut off and transition velocity) on per-servo basis.
- Stackable. You can connect multiple controllers to the same serial line, provided you've configured the servo numbers beforehand, all the way to 127 (254 in Mini SSC II compatibility mode).

2.2. Bad Things

Not a bad thing per se, but a result of a tradeoff with flexibility: protocol overhead is quite high, 6 bytes per positioning command. Well, given the high baud rate the controller can sustain, and the fact that you can optimize transitions with hardware supported constant velocity, this becomes rather a theoretical issue.

Note:

If you need to speed up the communications between the controller and your application, you have a choice of putting the controller into Mini SSC II compatibility mode (which requires 3 bytes per positioning command), but in this case you will lose the ability to use hardware supported constant velocity transition.

2.3. Features not available through Servomaster driver

Servomaster uses only three of 6 available Pololu commands: Command 0 (set parameters) to shut off an individual servo, Command 1 (set servo speed), and Command 4 (set position, absolute). It is not a deficiency, but a design decision - I thought that the rest of the functionality (detailed below) is better implemented with abstractions implemented in Servomaster itself, and not through hardware commands.

Here's why:

- Command 2 (set position, 7 bit): the result of this command's execution is stateful, it depends on the values passed before to Command 0 (set range). Whereas Command 2 takes 5 bytes, not 6, the overhead reduction is insignificant when compared to increase in driver complexity (the driver which takes `double 0..1` range will have to have internal logic to keep track of what state the controller is in).
- Command 3 (set position, 8 bit): it's still 6 bytes. No benefit whatsoever.
- Command 5 (set neutral): given the fact that Servomaster supports the notion of min/max range, this command seems superfluous.

Note:

Note that the unused functionaly seems superfluous only in the context where Servomaster makes sense - when the huge overhead of a JVM is not considered excessive. These unused commands do benefit simple designs.

2.4. Note on Stacked Configurations

According to documentation, Pololu controllers can be stacked (multiple controllers connected to the same serial port), and their servo numbers can be changed.

Servomaster currently doesn't support sparse servo arrays, so this feature is not explicitly supported. This is due to the fact that none of servo controllers supported report back what servos are connected and what aren't.

However, it is possible to support all the servos on all the controllers connected - all you have to do is change the value returned by `getServoCount()` in the implementation class.

If you need this feature, let us know - it's trivial to implement.

3. Serial Controllers

Can't say anything in particular until I get one, but it seems to me that getting one is a really good choice.

4. Dual Interface Controller

Due either to my stupidity or bad luck, I've managed to burn the USB part of it - the serial port still works. I guess it's been too long since I've done any real hardware work, save for putting together FT639. In any case, I have contacted Pololu, and they said they'll try and figure out the cause for this failure. Preliminary verdict is a bad USB/Serial bridge chip.

> **Note:**
> Keep in mind that you will need a level convertor for this serial port implementation - it uses TTL levels, not RS-232 levels. The schematics for this convertor are in the PDF user manual of Dual Serial Motor Controller. Plus, you'll have to pull the reset line to high - 4.7K resistor to +5V will do the job.

4.1. Good Things

- Mini USB connector, unlike USB B connectors used in other controllers;
- Convenient screw-in power connector.

4.2. Bad Things

- It is not a true USB device. You won't be able to benefit from generic features offered by USB interface, in particular, hotplugging. This device must be accessed as a virtual serial port (well, unless someone writes a direct driver that will treat this device as a servo controller, and not a serial port).
- The USB/Serial bridge chip used in this controller (CP2101 by Silicon Laboratories) is not supported on Linux (Google search on "CP2101 Linux" at the moment of writing

produces this link at the top), at least currently. SiLabs refused to provide the chip protocol specifications when I asked, and despite the fact that they promised to open the code a while later, it seems to be just a lip service - you can't really consider an obfuscated piece of C code with no comments Open Source, can you?

At the moment of writing, it appears that someone reverse engineered the CP2101 chip and produced a Linux kernel module. This is better than nothing, however, it is still questionable whether the module will work properly with RxTx. Plus, I have my usual reservations against writing kernel drivers where user mode driver will suffice.

If I have enough time and/or motivation later, I'll just rewrite the driver as a part of Servomaster, thus adding true USB functionality to this controller's driver.

- *[Just a wish]* One mounting hole may be problematic given the fact that there'll be at least one sturdy (USB) cable and 16 servo cables. Of course, any attempt to add another mounting hole will result in the PCB getting bigger, but one can always wish, right?

5. Conclusion

Getting a serial Pololu controller is a no-brainer - it is compact and elegant; go get it.

Getting a USB controller is asking for trouble until the direct USB driver is written. Even for Windows, where a binary driver is available from SiLabs, you won't get the advantages of USB hotplugging - and this may be a showstopper for a mission critical application.

6. Acknowledgements

I'm grateful for troubleshooting assistance from Pololu Corporation. Special thanks to Jan Malasek.