Heater Driver
Hardware & Serial Interface Manual

J. W. Silverstone

September 2013, Firmware 1.1

Contents

§1 Serial port configuration

Serial (RS-232) communication from the PC should be configured with the following parameters. These are default parameters on many systems, so this configuration may require no changes at all.

9600 bps | *1 stop bit* | *no flow control* | *no parity*

§2 COMMAND STRUCTURE

Instructions are sent in human-readable form, as a series of ASCII characters, terminated by a line-feed character (0x0A, '\n' in C). Note that PuTTY, by default, inserts only a carriage return (0x0D), which is ignored by this driver version. Instruction codes are *not* case-sensitive, so can be written in the user's preferred format. The `BACKSPACE` character is correctly handled, for ease of manual usage. The following conventions are used in the command set description (§3):

`[float]`	a positive floating-point value, ex "3.14159".
`[port]`	a (base-ten, zero-indexed) port address, ex "0", "07", "31415". The address of each port is counted from port 0 of the first device in a chain. Each device subtracts its number of ports from the address. If the resulting value is negative, then it handles the instruction, otherwise, it passes it down the chain.
`[bool]`	a boolean value, either true or false, "1", "0", respectively.
`[int]`	an integer value, ex "1", "05", or "23".

§3 INSTRUCTION SET

At any time, a list of instructions and their descriptions can be requested from the device, by sending the code `help`.

3.1 Output control

These instructions control the device outputs. In response to instructions which have no return value, the device will write to the serial bus `OK`, or an error code (§4), terminated by a line-feed character (0x0A).

`V[port]=[float]`	Set the voltage on `[port]` to `[float]` volts.
`I[port]=[float]`	Set the current sourced to `[port]` to `[float]` milliamperes. The target device enters a strongly overdamped control loop on the specified port, adjusting the output voltage to obtain the specified current.
`P[port]=[float]`	Set the power sourced to `[port]` to `[float]` milliwatts. Like `I[port]=[float]`, the device will control the power sourced to the selected port.

`Vall=[float]` Set the voltage on all ports to `[float]` volts. Equal to sending `V[port]=[float]` for each port.

`Iall=[float]` Set the current on all ports to `[float]` milliamperes. Equal to sending `V[port]=[float]` for each port.

`Pall=[float]` Set the power on all ports to `[float]` mW. Equal to sending `P[port]=[float]` for each port.

`Vmax[port]=[float]` Set the maximum software voltage (`VMAX`) on a given port, such that if the device receives an instruction `V[port]=[V > VMAX]`, it will instead execute `V[port]=[VMAX]`, and write an error to the serial bus.

`Imax[port]=[float]` Set the maximum current (`IMAX`) for a given port. If the device receives an instruction `I[port]=[I > IMAX]`, it will instead execute `I[port]=[IMAX]`. Additionally, if the current sourced by `[port]` exceeds `IMAX`, the device will implement a software "fuse", whereby the device will execute `V[port]=0.0`, write an error to the serial bus, and neutralise the port until further `V[port]=`, `I[port]=`, or `P[port]=` instructions are received.

3.2 Measurement

Each of these instructions requests a floating-point value from the device. In response to each valid request, the device writes this value to the serial bus, as `[float]`, terminated by a line-feed character.

`V[port]?` Measure the voltage on `[port]`, in volts. NB: devices Rev. 2 and earlier have no voltage measurement facilities, so only the programmed voltage is returned.

`I[port]?` Measure the current sourced by `[port]`, in milliamperes.

`P[port]?` Measure the power on `[port]`, in milliwatts.

`VIPall?` Measures and returns a listing of all voltages, currents, and powers on all ports, for every connected device.

3.3 System

What follows is a listing of miscellaneous codes, to obtain various system and help information from the device.

`help`	Writes a list of valid device instructions, with a short description of each, to the serial bus.
`ping`	Each device connected to the chain will respond with “ping", indicating how many devices are connected to the PC.
`echo=[bool]`	Turn off (`[bool]=0`) or on (`[bool]=1`) serial echoing.
`led=[bool]`	Turn off (`[bool]=0`) or on (`[bool]=1`) indicator LED operation.
`Vmax?`	Returns the device maximum voltage.
`version?`	Returns the device firmware version number.

§4 Error codes

In the event of an error, the device will write `ERR[int]:[port]` to the serial bus, where `[int]` represents an error code, and `[port]` is an optional port address which depends on the error code. For codes with no associated port, `[port] = 00`. Like instructions from the PC, all error codes are terminated with a trailing 0x0A character.

`ERR00:00`	An unknown error occurred.
`ERR01:[port]`	An over-voltage event occurred on `[port]`: a `V[port]=` instruction was received which violated `VMAX` on `[port]`. This port is reset to voltage-controlled mode, and 0 V.
`ERR02:[port]`	An over-current event occurred on `[port]`: either an `I[port]=` instruction was received which violated `IMAX` on `[port]`, or a current greater than `IMAX` was measured on that port. This port is reset to voltage-controlled mode, and 0 V.
`ERR10:00`	Unrecognised instruction received from PC.
`ERR11:00`	Invalid parameter within valid instruction from PC.
`ERR12:[port]`	Invalid port within valid instruction from PC.

§5 HARDWARE

The electronic hardware consists of two sets of circuitry: digital logic and communications, and analogue drivers and measurement. Logic is handled by a PIC24FV32KA304 microcontroller (MCU), which controls the flow of digital information into and out of the device, and directs the analogue circuitry accordingly. In the centre of the analogue circuits are a number of either high-precision (0–10V operation, TS922IDT) or high-voltage (0–20V operation, LM7322MA) operational amplifiers, together with various passive protection and filtering circuits. The analogue and digital sides are linked by a 12-bit precision, high-speed, internally-referenced digital-to-analogue converter (Texas Instruments DAC7678SPW).

Voltage actuation is realised via a signal from the MCU to the DAC, whose output is then amplified. Current measurement is facilitated by a 2–Ω shunt and differential amplifier in the output path of each port; this is sent back to the MCU for digitisation by one of eight onboard 12-bit precision analogue-to-digital converters (ADC).

5.1 Accuracy

Voltage accuracy is affected by a number of factors. The raw DAC output is accurate to within its stated precision (12-bit, or $5\text{V}/4096 \approx 1.2\text{mV}$), but this value is amplified to the required level and load-driving capability—with gain of either 2 or 4, for 10V or 20V operation, respectively. This implies an output precision of 2.4mV (10V mode) or 4.8mV (20V mode). Each amplifier is specified in terms of noise spectral density: while both models are low-noise, the 10V amplifier, at $9\text{V}/\sqrt{\text{Hz}}$ is somewhat lower than the 20V version, at $12\text{V}/\sqrt{\text{Hz}}$. Unfiltered, these values would lead to μs-long spikes as high as 9- or 12-mV, respectively. However, on-board filters with a cut-off frequency of 1kHz limit this noise source to a negligible $10\,\mu\text{V}\cdot\text{ms}$. The largest source of error, then, is not random, but uncalibrated systematic errors in the gain of each channel, which amounts to $3\sigma = 1.2\,\text{mV/V}$. This error is minimised due to amplifier design, but is nonetheless present, and must be accounted for in high-precision applications.

Current measurement accuracy is limited by random ADC sampling noise, which is measured at $\pm 150\,\mu\text{A} \pm 3\%$ per single current reading. For accurate current (or power) readings, use repeated measurements. The ADC readings are referenced to the 5V logic supply, and this could cause referencing errors. This design will be improved in a future revision.

5.2 Transient behaviour

To limit transient effects and associated risks, each output is designed be strongly over-damped. The input to each output amplifier is equipped with a low-pass RC filter, with $R = 100\,\text{k}\Omega$ and $C = 10\,\text{nF}$, with a corresponding cut-off frequency of 1 kHz. Additionally, each output is further smoothed by a high-voltage 1-μF tank capacitor. An example of device step response is presented in Fig. 1.

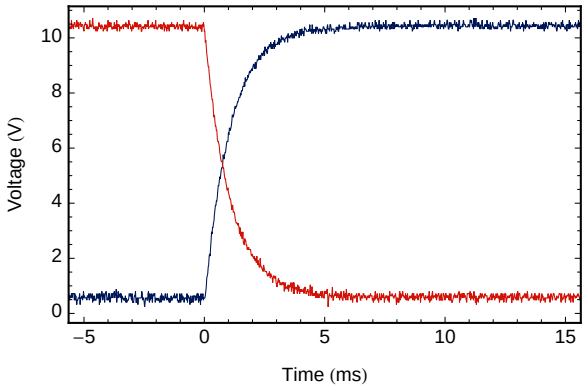


FIG. 1: Transient behaviour. Characteristic traces for rise (blue) and fall (red) step responses from 0 V to 10 V. Both exhibit decays of 2 dB/ms These were measured on Output 0 of Board 2 (Rev. 2, 0–20 V). The small voltage offset results from a design flaw in all Revision 2 boards, and can be corrected in software.

§6 DAISY-CHAINING

The drivers may be cascaded, or *daisy-chained*, from a single RS-232 port, and power supply, as demonstrated in Fig. 2. No extra software configuration is necessary, but the number of responding boards in a chain can be tested using the `ping` command (see §3.3).

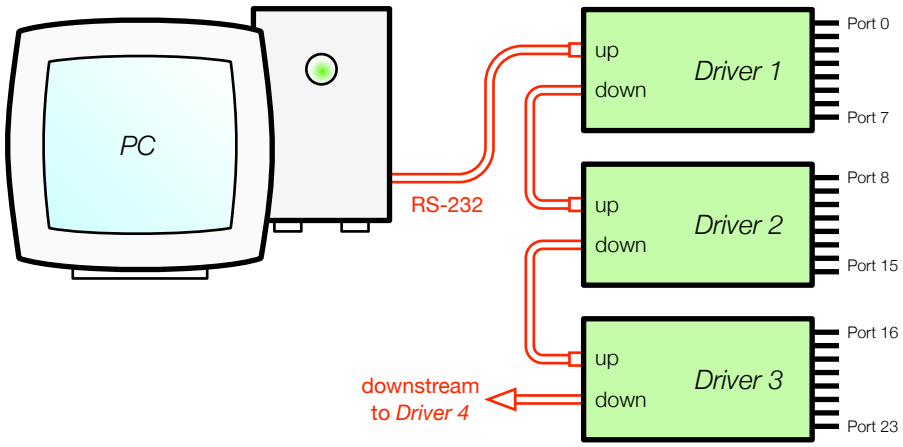


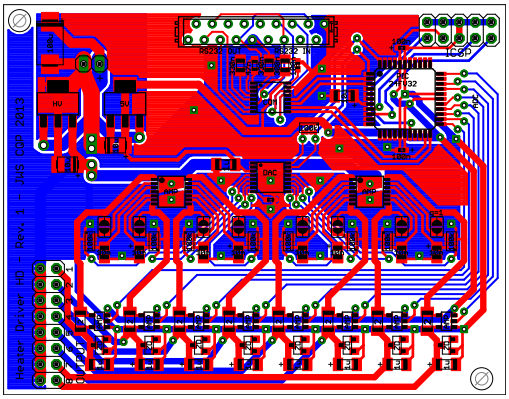
FIG. 2: Schematic configuration of three or more daisy-chained driver boards, as discussed in §6. Successive upstream and downstream ports are connected by serial cables, such that each device may pass commands downstream to the next in the chain. Ports are addressed starting from 0, nearest the PC, and increasing by 8 for each chained board, as shown.

§7 ACKNOWLEDGEMENTS

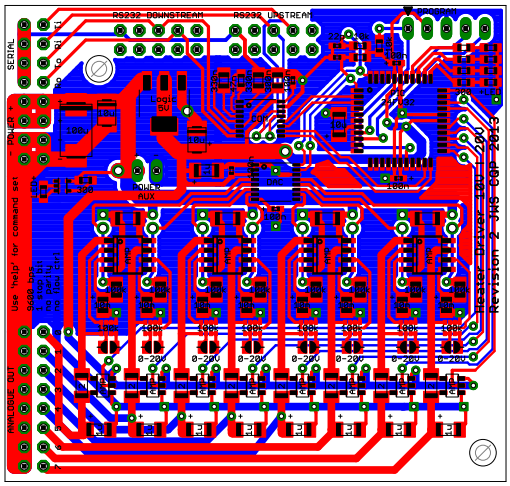
This work would not have been possible were it not for the following talented people. Special thanks to: **M. Piekarek** for extensive testing, debugging, commissioning, and assembly; and **M. Loutit** for essential assistance in early design, programming, and assembly.

§8 LIST OF REQUESTED FEATURES

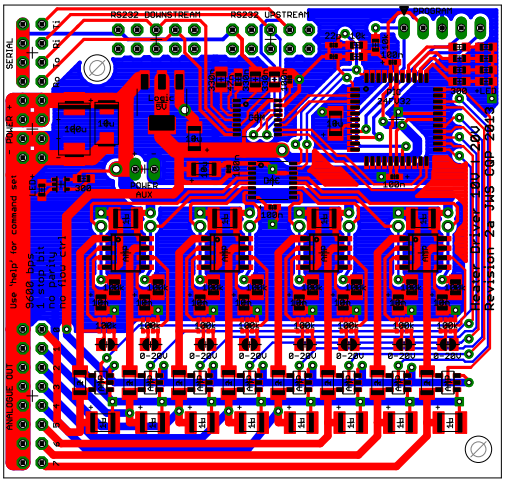
- Upstream/downstream symmetry
- Non-volatile calibration data
- Zero-offset outputs (zero voltage really means it)
- Simpler assembly
- Detailed usage instructions on reverse silkscreen



Revision 1



Revision 2



Revision 2a

FIG. 3: Board layouts. Collected above are layouts for each revision of the driver board to date. These can be used for identification, for reverse-engineering, or simply to see how the driver has evolved over time. Red traces are on the front of the board, while blue traces are on the back.