Logging Temp Controller v1

*If hardware (schematic and layout) and software have mutually dependent changes, use a new full version number. Use “dot” minor versions for functional changes without HW/SW interdependency. Use “double-dot” sub-minor for bug fix changes. E.g. a PCB layout bug fix to a v1 would number the PCB only as v1.0.1*

# Technical Issues

## Power

Assumed to be a 5V wall-wart regulated down to 3V3 because of SD card requirements. Power is NOT taken from USB, even when connected (the internal 3V3 regulator in a FT232R is not rated to deliver sufficient current for SD Card).

Two options for LDOs are accommodated. Only use one!

NB if programming with a FT232RL board, it MUST be set to 3V3 I/O.

## Solder Jumpers

SJ1: bypass LDO if 3V3 supply is used.

SJ2: select input to PD2, either a) use to get a square wave or alarm interrupt from the MCP7940, or b) use to sense a keypress on S4 or SD card insertion

SJ3: connect SD card insertion switch. S4 should be omitted in this case

Normal use expected to have SJ1 open, SJ2 to option (b) and SJ3 closed.

## Omitted Components

S4 – because of SD card switch

R2, R3 - when using the Arduino Wire library, ATMega internal pull-ups enabled in code. TWI pullups may be necessary with more I2C devices or when using 400kHz rate)

LED2, R8, JP4 – not expecting to control 2 outputs, these are “for future use”

SV1 – for possible extra I2C components “future use”

JP2 – don’t expect to use ISP

C7 – there is no ADC use. This is present “for future use”.

\*\* it may be possible to omit some/all of the 22k pull-ups on the SD Card lines but these seem to be generally recommended.

## LCD Notes

This is a 3V3 device. Some 3V3 versions have peculiar requirements for V0.

The LCD may be omitted. Place a 22k resistor from E to ground; the software will sense this and skip LCD output code.

## Watchdog

Unless #define WATCHDOG has been commented out, the watchdog timer is set to 8s and a delay of >8s between calling loop() will cause a reset. This is useful because if there is a crash while outA is switched on, the brew will overheat (or if off, over-cool!).

# User Notes

## Inserting and Removing the SD Card

Removing the card during a log write may cause problems but otherwise, it should be OK to remove and insert a card while the controller is operating. To be safe, do not remove in run mode. Newly acquired data is lost when no card is inserted (no buffering).

## Logging to CSV

The log data is largely self-explanatory, except that the columns after time, and before temperature are: outA, outB and the status byte. The status byte (currently) reads 0 except for the first log after a reset, when it indicates the cause of the reset (power-on, manual, watchdog). See below for outA and outB.

## Temperature

### Sensors

Maxim DS18B20 temperature sensors have 64 bit IDs but these are used internally and matched to a zero-based index in a lookup table is stored in EEPROM, where the 1st stored ID is "sensor 0" etc. On first startup, discovered sensors have their IDs stored to EEPROM in the order they are found by OneWire library. In future the same sensor will have the same index so to know up-front which probe will be 0, 1, 2 etc, you should connect one probe at a time with a reset/restart to cause it to be detected. Otherwise use the \*SWAP and \*SCAN commands. A look-up array is kept to record which sensors are actually attached (some may be disconnected but retain a record in the EEPROM).

### Control Rules

This is simple min/max system. The values of Tmin and Hysteresis are set via serial commands.

If T>=Tmin+Hysteresis, then output A is turned off. If T<Tmin then the output is turned on. In between Tmin and Tmin+Hysteresis, the output remains in the same state. The value is logged; a 1 means the heater is turned on.

Only ONE of the temperature probes is used to control the output. It will be the connected sensor at index 0 in the EEPROM table if that sensor is connected. Otherwise, it will be the first sensor detected (1-wire detection order).

Output B is not used for control but, in the current version, is used to light a LED to show that T>Tmin+Hysteresis. It will turn off again once the temperature drops to T<=Tmin+Hysteresis. i.e. output B signals a “too high” or over-shoot condition. The value is logged; a value of 1 means the temperature is above range.

Outputs only change in run mode (see below).

## Start-up Reset

The output B LED will light for 0.5s on startup.

Following startup a non-zero status value will be written to the first logged record. Depending on the optiboot version[[1]](#footnote-1), the status byte indicates the cause by a status byte in which the high nibble is F and the low nibble is given by the MCUSR register:

* Bit 3 – WDRF: This bit is set if a Watchdog System Reset occurs
* Bit 2 – BORF: This bit is set if a Brown-out Reset occurs.
* Bit 1 – EXTRF: This bit is set if an External Reset occurs.
* Bit 0 – PORF: This bit is set if a Power-on Reset occurs.

~~If the pushbutton (S2) is held closed during power-on or manual reset then any previously-stored temperature sensor IDs (in EEPROM) are cleared before scanning for new attached sensors.~~

## Modes

S5 selects “set” or “run” mode.

In set mode:

* serial commands may be sent (see below) and the push button (S2) may be used to cycle the set-minimum temperature in 0.5C increments from 0 to 30C.
* Changes made in set mode are applied immediately to the real time clock but other settings are saved to EEPROM only on return to run mode.

In run mode:

* Serial commands are ignored
* The push button (S2) may be pressed to cause the display to cycle through the temperature readings according to all attached probes. During this time, output control is suspended. Holding the button for more than 8s will cause a reset (which is harmless).

## Serial Commands – Setup and Testing

Send \* followed by a four letter command, optionally followed by an argument (no spaces). Send CR after the command. (LF will be ignored but it causes a short delay, so it is best not sent).

ECHO echo test – follow the command with a string

GETD get date string

SETD set date using YYYY-MM-DD string

GETT get time string

SETT set time using HH:MM:SS string

LOGN change name of log file. Must be no longer than 8.3 and must be at least 1.3

LOGI change logging interval in seconds. this is determined by arduino function millis() not the RTC. An integer

TEMP read all attached temp sensors and emit logging string. Forces a check of the control conditions and change of output

TMIN change low temperature threshold for controller. output turns on if T less than this value. May be a float. Maybe negative

THYS change temperature hysteresis. output turns off if T >= TMIN+THYS. May be a float

OUTI change interval in seconds between controller deciding whether to switch on or off (according to TMIN and THYS). An integer

OUTS read the output status over serial

RELD clear stored values in EEPROM. This stops the device and must be followed by a hard reset.

SWAP swap temp sensor IDs in EEPROM and re-scan. Argument is the single digit sensor index (1,2...) to swap with the sensor ID at index=0.

SCAN clears IDs stored in EEPROM and does a rescan of attached sensors and reports the IDs via Serial. NB if more than one sensor then it may be necessary to use \*SWAP so that the desired sensor is used for control

E.g. send “\*ECHO” or “\*GETT” or “\*SETT12:43:21”

1. The high nibble will always be set after startup. If the optiboot version is late enough, the low nibble will be correct, otherwise it is probably indeterminate (but has been observer to be 0xF) [↑](#footnote-ref-1)