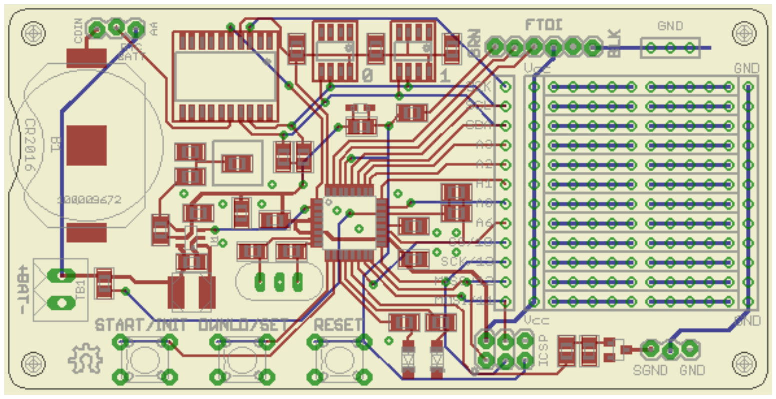
Double-A Data Logger
User's Guide and Technical Reference

Version 1, April 2013

Hardware v2

Software v1



CC BY-SA

Double-A Data Logger by Jack Christensen, including this User's Guide and other documentation, hardware and software, is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License.

To view a copy of this license, visit:

http://creativecommons.org/licenses/by-sa/3.0/

Or send a letter to:

Creative Commons
Suite 300
171 Second Street
San Francisco, California, 94105, USA

Table of Contents

1 Overview

The Double-A Data Logger is an open source, Arduino compatible, low power, battery operated portable data logger. It is primarily aimed at low-rate (once per minute or less), long-term logging such as environmental monitoring or similar scenarios. Its features include:

- ATmega328P-AU microcontroller (as used in the Arduino Pro and Pro Mini).
- 256k or 512k bytes EEPROM data logging memory. With respect to power consumption, EEPROM compares favorably with flash memory or SD cards, has an endurance of more than 4 million write cycles, and data retention of more than 200 years.
- An accurate real-time clock (±2ppm accuracy from 0°C to +40°C) with 236 bytes battery-backed SRAM and integrated digital temperature sensor with ±3°C accuracy.
- Power supplied by two AA alkaline cells (not included), which can power the logger for up to a year.
- Prototyping area for sensors.
- Pins available for sensors:
    - Four digital pins: D10-D13 (SPI bus: SS, MOSI, MISO, SCK)
    - Four analog pins: A0-A3 (PC0-PC3)
    - I2C bus A4-A5 (SDA, SCL) shared with EEPROM, RTC
    - A6-A7 (PC6-PC7) are connected to a voltage divider dedicated to measuring battery voltage
- Boost regulator to provide 3.3V (5V optional) for sensors and peripheral devices.
- Less than 5µA current consumption while sleeping.
- Complete code for logging, downloading data, setting the RTC, etc., that can be modified to interface to various sensors.
- Compact size, 3.90 x 1.95 inches.

1.1 Prerequisites

The following are required to use the Data Logger:

- A USB-to-TTL serial converter (e.g. FTDI) to download data from the logger to a computer.
- An in-circuit serial programmer (ICSP) for programming the logger. To save power, the logger code adjusts the microcontroller's system clock frequency; therefore a bootloader is not used.
- Experience with Arduino and C++ programming in order to implement code for sensors.

2 Quick Start

The Double-A Data Logger is provided with logging software that records the following data items once per minute:

- Date/Time
- Temperature (from the sensor in the real-time clock)
- Battery voltage
- Regulator voltage

Before adding sensors and associated code, take a few minutes to become familiar with the logger using the provided software.

2.1 Prepare the logger

1. Connect the battery box to the terminal block on the logger. **Be sure to observe proper polarity: Red wire to plus (+), Black wire to minus (–).** *Reverse polarity will damage the logger!*

2. Insert two AA cells into the battery box, *again being careful to observe proper polarity*. For long-term logging, only fresh AA alkaline cells are recommended. NiMH or NiCd cells can also be used, but these will have shorter life, and are ***not*** recommended if temperature extremes are expected. Consult the datasheet or product guide for the cells being used and ensure that they are operated within specifications.

3. Using a USB-to-TTL serial converter attached to the six-pin connector labeled FTDI, connect the data logger to a computer running a terminal program at 57600 baud. Observe the orientation of the connector (BLK/GRN) carefully.

4. Press the RESET button briefly and release it, the logger should respond with a message similar to:

```
Double-A Data Logger SW-v1
02Apr2013 02:36:22 UTC
01Apr2013 22:36:22 EDT

512kB EEPROM, 0 Records logged, 100.0% Available.
Log interval 00:01:00, Record size 12 bytes, NO-WRAP mode.
```

5. If the RTC is not running, e.g. the backup coin cell has just been inserted, the time must be set before proceeding. Press RESET again and check the time to verify that the RTC is running. If not, see the section, Setting the real-time clock.

2.2 Initialize and start logging

6. At this point, the logger is in the COMMAND state, as indicated by the blinking red LED. It will remain in COMMAND state for 30 seconds, and then if no further buttons have been pushed, it will go into POWER DOWN state.

7. If the logger has powered down, press RESET again to wake it.

8. Press and hold the START/INIT button for two seconds until both the red and green LEDs blink three times, indicating the logger has been initialized. It should respond with a message similar to:

```
INITIALIZED

512kB EEPROM, 0 Records logged, 100.0% Available.
Log interval 00:01:00, Record size 12 bytes, NO-WRAP mode.
```

9. Press the START/INIT button briefly and release it to begin logging. The logger should respond with the following, and the data will appear as it is logged, once per minute by default:

```
LOGGING
00:43:00 05May2013, 756, 2681, 3332
00:44:00 05May2013, 815, 2681, 3332
00:45:00 05May2013, 779, 2681, 3332
00:46:00 05May2013, 765, 2681, 3332
00:47:00 05May2013, 824, 2681, 3332
```

2.3 Download the logged data

10. After logging several records, press RESET to stop logging and return to COMMAND state.

11. Press the DWNLD/SET button briefly and release it to download the logged data. It should appear similar to the following. Note that the temperature is Fahrenheit times ten, and the voltages are in millivolts. Further, even though the logger only records the timestamp as UTC, it is output as both UTC and local time with a time zone abbreviation. The data is in comma-separated-variables (CSV) format:

```
utc,local,tz,tempRTC,vBat,vReg
2013-05-05 00:43:00,2013-05-04 20:43:00,EDT,756,2681,3332
2013-05-05 00:44:00,2013-05-04 20:44:00,EDT,815,2681,3332
2013-05-05 00:45:00,2013-05-04 20:45:00,EDT,779,2681,3332
2013-05-05 00:46:00,2013-05-04 20:46:00,EDT,765,2681,3332
2013-05-05 00:47:00,2013-05-04 20:47:00,EDT,824,2681,3332
```

12. At this point, logging can be resumed (retaining the previously logged data) by pressing START/INIT briefly, or the logged data can be discarded by initializing again. Then logging can be started again, or the logger can be allowed to power down.

3 Detailed Operation

The Double-A Data Logger is state machine controlled by the three buttons on the board:

- START/INITIALIZE
- DOWNLOAD/SET
- RESET

Two LEDs, red and green, give feedback to indicate the state of the logger.

3.1 Command state

Press and release the RESET button to enter the COMMAND state. This will wake the logger from SLEEP or POWER DOWN state, and stop logging if in progress. COMMAND state is indicated by the red LED blinking once per second.

If no commands are given by pressing additional buttons within 30 seconds, the logger will go into the POWER DOWN state to save power. No logging occurs in the POWER DOWN state. Pressing RESET to enter COMMAND state again is the only way to leave the POWER DOWN state.

3.2 Initializing the logger

Before first use, or whenever the logging parameters (data fields, logging interval, etc.) are changed[1], the logger must be initialized. From the COMMAND state, press and hold the START/INIT button for two seconds. Both the red and green LEDs will blink three times to indicate that initialization is complete.

Initialization does not erase the EEPROM logging memory, but just resets the EEPROM pointer to begin logging at the bottom of the EEPROM address space.

3.3 Setting the real-time clock

Using a USB-to-TTL serial converter attached to the six-pin connector labeled FTDI, connect the data logger to a computer running a terminal program at 57600 baud. Observe the orientation of the connector (BLK/GRN) carefully.

From the COMMAND state, press and hold the DWNLD/SET button for two seconds. The green LED will begin blinking once per second to indicate that the logger is waiting for input. Using the terminal program, enter the current Coordinated Universal Time (UTC) in the format `yy,m,d,h,m,s,` and send it to the logger. Use a 24-hour clock for the time. Entering a terminating comma after the seconds will avoid Arduino's one-second Serial timeout and allow the time to be set more accurately. The logger will respond with the date and time the RTC was set to.

[1] See Software Reference for information on changing the logging parameters.

It is strongly recommended that the RTC always be set to UTC. If local time is desired when downloading logged data, the conversion should be done during the download process. The provided logging code shows how to do this.

The current UNIX epoch time can be found at http://www.epochconverter.com/.

When in the SET state (green LED blinking), the logger will wait 30 seconds for input, and if not received, will enter the POWER DOWN state.

3.4 Logging

Once the logger is initialized and the RTC set to the current UTC time, logging can be started by briefly pressing and releasing the START/INIT button. The green LED will blink three times to indicate that logging has begun. As the first five records are logged, the green LED will briefly blink for each. In between logging each record, the logger enters a SLEEP state to conserve power.

Logging can be stopped by pressing RESET to enter COMMAND state. At that point, the logged data can be downloaded, the logger can be initialized, or logging can be resumed by briefly pressing and releasing START/INIT.

Once started, the logger will continue logging until RESET is pressed, or until the EEPROM memory is full. Once memory is full, the logger will go into POWER DOWN state, unless it is set for wrap mode, in which the oldest data is overwritten with the newest data. Wrap mode is an option that is set in the software; recompiling and loading the software to the logger is required to change the mode.

3.5 Download the logged data

Stop logging and enter COMMAND state by pressing RESET.

Using a USB-to-TTL serial converter attached to the six-pin connector labeled FTDI, connect the data logger to a computer running a terminal program at 57600 baud. Observe the orientation of the connector (BLK/GRN) carefully.

From the COMMAND state, briefly press and release the DWNLD/SET button. While the data downloads, the red and green LEDs blink alternately. When the download is complete, the logger returns to the COMMAND state.

Use the features of the terminal program to capture the downloaded data to a file, or simply use copy and paste.

Logging can now be resumed, retaining the previously logged data, by briefly pressing and releasing the START/INIT button. Alternately, the logger can be initialized to discard the logged data.

4 Technical Reference

This remainder of this guide discusses the hardware and software features of the Double-A Data Logger that should be understood before modifying or extending its capabilities.

4.1 Open source

The Double-A Data Logger is an open source project. Source documents can be found at:

- http://github.com/JChristensen/aaLogger_HW (Schematic and board)
- http://github.com/JChristensen/aaLogger_SW (Software)

4.2 Design overview

The primary design consideration for the Double-A Data Logger was low power consumption, in order to achieve long battery life. To this end, key design elements include:

- A small boost regulator to provide regulated voltage when required, but that can also be put into a shutdown/bypass mode in which it draws negligible current, and passes the battery voltage directly through to the circuit.
- Dynamically reducing the microcontroller's clock frequency to reduce power consumption and also putting it into low-power sleep mode in between logging samples.
- Use of EEPROM for logging memory rather than flash memory or SD cards.
- Independent control of power to the on-board peripherals (EEPROM and RTC) and sensors.
- Operation on a 3.3V supply. While there is an option (via a solder jumper on the board) to operate at 5V, the only reason to do so would be if a sensor is used that cannot operate on the lower voltage.

4.3 Typical logging cycle

The microcontroller controls the regulator boost/bypass mode as well as power to on-board peripherals (EEPROM and RTC) and sensors. While the provided logging software demonstrates this, a typical operational cycle is as follows.

Note that while the MCU uses an 8MHz crystal for its system clock, the CKDIV8 fuse is also programmed in the fuse bytes, therefore the system clock is effectively 1MHz after a reset or power-up. This ensures that the MCU is not over-clocked down to its minimum operational voltage of 1.8V when the boost regulator is not enabled.

1. An alarm signal from the RTC generates an interrupt to the MCU, which brings it out of low-power sleep mode.
2. The MCU enables the boost regulator, increasing the supply voltage to 3.3V (or 5V) and increases the system clock to 8MHz.

3. The MCU enables power to the sensors, reads the sensors, and stores the sensor data in the log data structure.
4. The MCU disables power to the sensors and enables power to the peripherals (EEPROM and RTC).
5. The MCU writes the sensor data to EEPROM and sets the next RTC alarm.
6. The MCU disables power to the peripherals, reduces the system clock to 1MHz, puts the regulator into bypass mode, and goes to sleep.
7. The cycle repeats when the next RTC alarm occurs.

5 Hardware Reference

5.1 Battery and regulator

The logger and its on-board peripherals (EEPROM and RTC) are designed to run on either 3.3V or 5V. To maximize battery life, 3.3V is recommended unless 5V is required for a sensor. If 5V is used, then either two or three AA cells can be used to power the board; *the regulator input voltage must be less than its output voltage for it to function correctly*.

There is a solder jumper on the board to configure the regulator to supply 5V if needed.

Note that the boost regulator is small and it should not be called upon to deliver more than about 100mA. With no sensors installed, the board should draw less than 10mA at 3.3V and less than 12mA at 5V.

5.2 Switched peripheral power

The EEPROM and RTC have modest current requirements, and are therefore powered directly from a microcontroller GPIO pin.

5.3 Switched sensor power

The prototyping area has rails for V_{CC} and ground (GND). Using a three-pin header and shorting block (or a jumper wire that can just be soldered in its place), the GND rail can be connected either to the true circuit ground, or to a switched ground (SGND) controlled by a low-side MOSFET switch controlled by the MCU.

Using the switched ground is recommended to minimize power consumption during sleep intervals. The MOSFET is a logic-level part, with maximum $R_{DS(ON)}$ of 27mΩ @ V_{GS}=2.5V, which will ensure a very small voltage drop across the MOSFET when it is on.

5.4 RTC backup battery selection

The RTC requires a backup battery to retain its timekeeping functions when the main supply voltage is off. This can be supplied either by a CR2016 coin cell, or directly from the main AA battery. Pads for a three-pin header (or just a soldered wire jumper) are provided on the board to make the selection.

Powering the RTC from the main AA battery instead of a coin cell will require setting the RTC time each time the main battery is replaced.

5.5 Installing EEPROMs

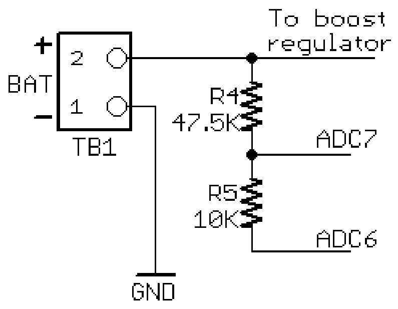
The logger uses M24M02 EEPROMs from ST Microelectronics. One or two EEPROMs can be installed, for a total storage capacity of 256k bytes or 512k bytes.

If only one EEPROM is installed, it must be installed in the position marked with a zero (0) on the board, which is part U4.

The software must be aware of how much EEPROM is installed; see the config.h section for details.

5.6 Measuring battery voltage

A voltage divider circuit as shown is provided for the purpose of measuring the battery voltage. It is intended to be used with the internal 1.1V ADC reference voltage. Note that the bottom leg of R5 (ADC6) will need to be connected to ground to make the measurement, but should be floated afterwards to avoid the current drawn by the voltage divider. The switched sensor ground described above can be used, or a GPIO pin can be used if one is available. ADC6 measures the voltage drop across the MOSFET or GPIO pin; this may be zero or nearly so, especially if the sensors draw a relatively small current.



To calculate the battery voltage in millivolts, assuming the 1.1V ADC reference is used:

$$V_{BAT} = \frac{1100}{1024}\left(\left(\frac{R_4 + R_5}{R_5}\right)(ADC7 - ADC6) + ADC6\right)mV$$

5.7 Power management

To ensure minimum power consumption and longest battery life, the user should read the ATmega328P datasheet, especially the *Power Management and Sleep Modes* section, and the *I/O Ports* section, and observe the recommendations therein.

6 Software Reference

As mentioned previously, the provided logging software logs date/time, RTC temperature, battery and regulator voltage once per minute. This section describes the modules (files) that will need to be changed to adapt the software for additional sensors.

6.1 config.h

This module contains the parameters that are most likely to require changing:

LOG_INTERVAL – Defines how often data is logged, in minutes. Must be between 1 and 60.

EEPROM_SIZE – Gives the total capacity in kilobytes (kB) for all installed EEPROM devices. Assuming M24M02 EEPROMs, it should be set to 256 if one EEPROM is installed or to 512 if both EEPROMs are installed.

EEPROM_PAGE – Gives the EEPROM page size in bytes and should always be 256 for M24M02 EEPROMs.

WRAP_MODE – If set to **false**, logging will stop once the EEPROM memory is full, and the logger will go into power down mode. If set to **true**, once memory is full, the logger will continue logging and overwrite the oldest data with new data. Logging will only stop if manually reset, or if the battery is exhausted.

6.1.1 LOG DATA STRUCTURE

config.h also contains the definition for the log data structure. Simply define variables within the struct braces {} as required.

To reference the variables in the structure, use **LOGDATA.fields.*variableName*** where *variableName* is the name given in the struct definition.

When using M24M02 EEPROMs, the size of the struct should be a multiple of four bytes if at all possible. This will minimize the number of write cycles and therefore maximize EEPROM endurance. Pad the struct out with a byte array of suitable length if needed, e.g. **byte RFU[n];**

6.2 logData.cpp

The logData::download() method will need modification to match the log data structure. See the comments in the module to locate the specific areas to change.

Also in logData::download(), as an example, the provided logging software outputs the date/time stamp both as UTC and as local time.

6.3 _main

The logSensorData() function will need modification to match the log data structure. See the comments in the module to locate the specific areas to change.

For time zone conversions and daylight/summer time changes, the continental US time zones are defined near the top of the module. Use the appropriate **TimeChangeRules** in the **Timezone** object, **myTZ**. Additional **TimeChangeRules** can be defined as needed. See the Timezone library for details.