Manual for aDCLoad

version 2.6

Written by F1RMB, Daniel Caujolle-Bert

Contents

Chapter 1

Arduino Programmable Constant Current Power Resistance Load

1.1 Original README.md from Lee Wiggins

This is all of the code, datasheets and design files for `my instructable`

- Arduino - It contains the Arduino code that we will be talking about here, within the dummy load folder. It also contains all of the 3rd party libraries I have used.
- Datasheets - It contains all of the datasheets for the major components used within the project.
- DesignSpark - I have used the opensource schematic and PCB design software for this project, its a fantastic free tool that has no limitations and I find it easier to uses than Eagle - http://www.rs-online.-com/designspark/electronics/eng/page/designspark-pcb-home-page The rev 1 folder contains all of my initial designs, please don't use this as there are 2 or 3 errors in the footprints plus I have completely revised the layout for rev 2, please only use these files. the gerber files are in there should you wish to have your own board done. See the next step for more information on this.
- LTSpice - This contains all of the LtSpice files from simulating the operation of the MOSFET.

Please checkout my instructable as it describes all of this code and the operation of the dummy load.

1.2 Informations on this code from Daniel Caujolle-Bert

Todo write that!

notes on Code::Blocks, flashing with avrdude/Makefile, provided HEX file, and so on

Chapter 2

ATmega32U4 fuses settings

Unlike the Arduino™ Leornardo board, the ATmega32U4 MCU used in this DC Load needs some special fuses settings.

The following command line defines them to the correct values:

```
avrdude -F -p atmega32u4 -C /etc/avrdude.conf -v -e -V -c usbasp -P usb -U lfuse:w:0xFF:m -U hfuse:w:0xD1:m
       -U efuse:w:0xCB:m -F
```

You can also invoke the provided *Makefile*, as:

```
make fuses
```

Chapter 3

Hardware modifications for version 2.6

Since the **Pulse Transient Time** and the **Input Relay** features introduction, few small hardware modifications are required:

- The LCD cabling (**!!! on the LCD's PCB only !!!**) should be modified as below:
  - resolder the cable from pins **d0**, **d1**, **d2** and **d3** to **d4**, **d5**, **d6** and **d7**, accordingly,
- Build the small input relay PCB (files available with source code),
- Get **GND** and **+5V** from the LCD connector (on the DC Load board, first and second pins), and connect them to the relay's PCB,
- Get the **+12V** DC from your power supply,
- connect old **d4** and **d5** (from the DC Load board LCD connector) to the relay PCB as **Relay** and **Button**, accordingly.

Chapter 4

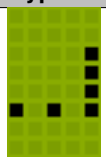
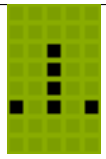
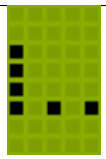
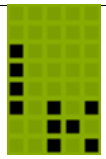
User interface overview

- The DC load control is done using a simple rotary encoder, which integrates a push button (see Extra control since v2.6 below for 2.6 specific enhancement).
- When the DC Load displays the input value (left aligned values), a single encoder detents turns the DC Load's display in settings mode (right aligned values), without any setting value changes.
- There are two display modes, **input values** and **settings values**.

  When you rotate the encoder, the DC Load switches automatically to **settings mode**.

  You just need to rotate the encoder to define the desired value, accordingly to the focus : **Current**, **Power** or **Pulse Transient Time**.
- In both display modes, a double click changes the focus (delimited by '**[**' and '**]**' symbols) to the next value parameter, Current (**I**), Power (**P**) or Pulse Transient Time (**t**).
- A simple click changes the tuning step.

  Next to the '**]**' delimiter symbol, an icon displays the tuning step multiplier, as following:

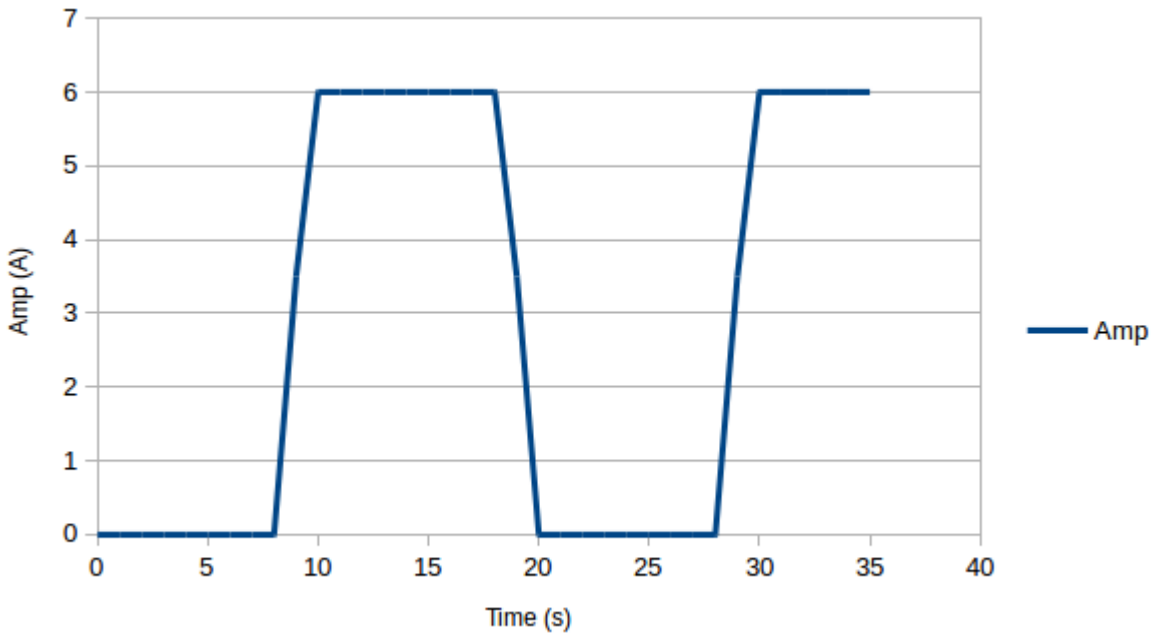
Multiplier	Glyph
x1	
x10	
x100	
x1000	

- By default, the DC Load displays **Input Voltage**, **Current load**, **Power dissipation**, **Pulse Transient Time** values and **heatsink temperature**.

Note

The Voltage is measured on the input connectors of the DC Load and may differs from the measured value out of the power supply source.

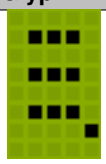
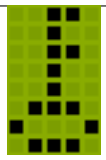
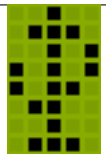
- The **Pulse Transient Time** permits you to define a pulse duration, from **0mA** to the defined loading current value. After this peak, the DC Load will switch to **0mA** loading current, for same duration. This will cycle endlessly until you set the **Pulse Transient Time** to **0s**.

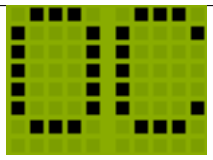
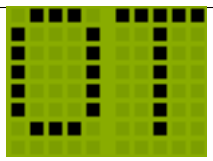


Note

The maximum duration is **8.192s**

- After 3 seconds in settings mode, without any encoder action, the DC Load returns to the **input values** display mode.
- According to the actual status of the DC Load, some icons may be shown:

Feature	Glyph
Logging is running	
Encoder is locked	
USB remote controlled	

Over-Current alam	
Over-Voltage alarm	
Over-Temperature alarm (in place of "℃")	

- To access to the options configuration, you need to press the button for more than 3 seconds. In this *window*, you can enable or disable the **backlight's auto-dimmer** and the **rotary encoder's auto-lock** features.

  A double click changes the option focus, a simple click changes the option enability and a long press exits the options *window*.

- When **auto-lock** is turned on and triggered, a double click unlocks the rotary encoder (the key icon disappear).

- When **auto-dimmer** is turned on and triggered, any rotary encoder action will turn the backlight on, without any change to the defined settings.

- There are 3 differents kind of alarms:

  1. **OC** for over-current:
     When **Over-Current** is triggered, Current setting is defined to 0mA, **OC** icon is displayed.
     Over-Current alarm will be cleared once the encoder is used to set a new Current value.
  2. **OV** for over-voltage:
     When **Over-Voltage** is trig<gered, Current setting is defined to 0mA, **OV** icon is displayed.
     Encoder will have no action until the input voltage drops below to its maximum value (24V).
  3. **OT** the over-temperature:
     When **Over-Temperature** is triggered, Current setting is defined to 0mA, **OT** icon is displayed.
     Encoder will have no action until the internal temperature drops below to its maximum value (80℃).

- The DC Load can be remotely controlled, see Remote Commands

4.1 Extra control since v2.6

- Since version 2.6, the DC Load uses a mechanical input relay. It's driven by a push button or a remote command. This permits to isolate the DC Load and the DUT. On any alarm, the input relay will disengage the DUT, to keep both devices in a safe state.

See Hardware modifications for version 2.6

Chapter 5

Remote Commands

See also Logging data format.

Note

Serial port configuration: **57600**,**8**,**N**,**1**

Warning

Commands and arguments are **case sensitive**, **ALL** in **UPCASE**

5.1 Get Identification

- **:∗IDN?:**
  - Returns firmware informations

See Return value.

5.2 Current setting getter

- **:ISET?:**
  - Returns current setting (in **mA**)

See Return value.

5.3 Current setting setter

- **:ISET:*value***
  - Set current ***value*** (in **mA**)

See Return value.

5.4 Calibration values getter

- **:CAL?:**
  - Returns the stored calibration values for **V**, **C**, **D** or **VD**.

See Return value.

See Calibration.

5.5 Calibration

- **:CAL:*toggle***
  - Turns ***ON*** or ***OFF*** the logging feature.
- **:CAL:*section*:*slope,offset***
  - ***section*** could be **V**, **C**, **D** or **VD**, standing for **V***oltage*, **C***urrent*, **D***AC* and **V***oltage* **D***rop*.
  - ***slope*** and ***offset*** are floating point values, with US period decimal separator ('.'). These values could be calculated using the *LibreOffice*'s spreadsheet file *aDCLoadCalibration.ods*.
- **:CAL:SAVE**
  - Backup calibation datas into EEPROM.

See Return value.

See Calibration Process.

5.6 DAC value setter (calibration purpose)

- **:DAC:*value***
  - Set DAC value (from **0** to **4095**).

Note

This command has no effect outside calibration mode

See Calibration.

See Calibration Process.

5.7 Current readed getter

- **:I?:**
  - Returns current readed from the load (in **mA**)

See Return value.

5.8 Voltage readed getter

- **:U?:**
  - Returns voltage readed from the load (in **mV**)

See Return value.

5.9 Logging enability

- **:LOG?:**
  - Printout if logging is ***ON*** or ***OFF***.

See Return value.

See Logging data format.

5.10 Logging enability

- **:LOG:*toggle***
  - Turns ***ON*** or ***OFF*** the logging feature.

Note

If ***toggle*** value is not specified, a single logging line is returned.

See Return value.

See Logging data format.

5.11 Pulse value getter

- **:PUL?:**
  - Returns pulse time value (in **ms**)

See Return value.

5.12 Pulse value setter

- **:PUL:*value***
  - Set pulse time ***value*** (in **ms**)

See Return value.

5.13 Input Relay status getter

- **:INP?:**
  - Printout if the input relay is ***ON*** or ***OFF***.

See Return value.

See Hardware modifications for version 2.6.

5.14 Input Relay status setter

- **:INP:*toggle***
  - Turns ***ON*** or ***OFF*** the input relay.

See Return value.

See Hardware modifications for version 2.6.

5.15 Return value

:*value*:*status*:

- Where:
  - ***value*** if any expected. **INVALID** on unknown command.
  - ***status*** could be **OK** on success or **ERR** on failure.

Chapter 6

Logging data format

See also Remote Commands

Note

fields are comma separated

6.1 CSV logging format

***timestamp*,*voltage*,*current sets*,*current read*,*temperature*\r\n**

- Where:
  - ***timestamp*** in **hundred of milliseconds**,
  - ***voltage*** in **mV**,
  - ***current sets*** in **mA**,
  - ***current read*** in **mA**,
  - ***temperature*** in **Celcius degrees**.

Chapter 7

Calibration Process

- **Prerequisites**:
  - **Hardware**:
    - Amp-meter,
    - Volt-meter,
    - Power supply (**0..24V**, **8A**)
  - **Software**:
    - A serial terminal emulator (e.g. "*HyperTerminal*" or "*Tera Term*" on Windows, "*minicom*" or "*cutecom*" on Linux).
    - The calibration spreadsheet file **aDCLoadCalibration.ods**
    - A software able to open the calibration spreadsheet, like "*LibreOffice*", "*OpenOffice*" and so on. The serial communication settings are: **57600**, **8**, **N**, **1**

- **Process Description**:
  - **Step 1: Maximum Current**

    Select "*Step 1*" tab in the calibration spreadsheet file.

    Connect the amp-meter and the power supply to the DC load, for current measurements. Open your serial terminal emulator, connect the DC load, then type:

    ```
    :CAL:ON
    ```

    ```
    :DAC:4095
    ```

    Write down the **mA** value readed on the amp-meter to the "mA_{max}" column. Now, type:

    ```
    :DAC:0
    ```

    Edit the aDCLoad.h source file, browse down the file, looking for the following line:

    ```
    static const float         CURRENT_MAXIMUM             = 7.845;     ///< Maximum value of
           load current (A)
    ```

    If necessary, change the 7.845 value to the one you've got on your amp-meter (don't forget to convert it from **mA** to **A**), then reflash the board with new code (using *Code::Blocks IDE* or the provided *Makefile*, running "*make burn*" command).

    Remember, if you have to reflash the board, that could be only done using ICSP programming. There is no bootloader flashed on the MCU, due to flash space restriction.

    Calibration step 1 is now done.
  - **Step 2: Voltage**

    Select "*Step 2*" tab in the calibration spreadsheet file.

    Connect your power supply to the DC load, sets to **0V**. The DC Load should the sets to **0mA**. If you've reflashed the firmware or didn't go through the step 1, then, in the serial terminal emulator, type:

    ```
    :CAL:ON
    ```

Set your power supply voltage output for each value in "V_{set}" column, and write down the readed value in "V_{read}" column.

Once you went through the whole array, the calibration string should be entered into the serial terminal emulator, like:

```
:CAL:V:x.xxx,y.yyy
```

Please note that the decimal separator **HAS TO BE** a period ('.'), as in US format.

Calibration step 2 is now done.

– **Step 3: Current**

Select "*Step 3*" tab in the calibration spreadsheet file.

Connect the amp-meter and the power supply to the DC load, for current measurements. Sets the output voltage to **5V**.

Using the DAC command, try to adjust its value to match each value in the "***A Amp-Meter***" column, and write down the readed value, on the LCD or serial terminal emulator output, into the "***A LCD/Term.***" column.

You can change the values in the "***A Amp-Meter***" column to strictly match the ones you're reading on the amp-meter.

The DAC command syntax is :

```
:DAC:value
```

where value is an integer from 0 to 4095.

Once you went through the whole array, set DAC value to **0**:

```
:DAC:0
```

The calibration string should be entered into the serial terminal emulator, like:

```
:CAL:C:x.xxx,y.yyy
```

Please note that the decimal separator **HAS TO BE** a period ('.'), as in US format.

Calibration step 3 is now done.

– **Step 4: DAC**

Select "*Step 4*" tab in the calibration spreadsheet file.

Connect the amp-meter and the power supply to the DC load, for current measurements. Sets the output voltage to **5V**.

Set the DAC value for each value in "***Steps***" column, and write down the readed value on the amp-meter into the "mA_{read}" column.

The DAC command syntax is :

```
:DAC:value
```

where value is an integer from 0 to 4095.

Once you went through the whole array, set DAC value to **0**:

```
:DAC:0
```

The calibration string should be entered into the serial terminal emulator, like:

```
:CAL:D:x.xxx,y.yyy
```

Please note that the decimal separator **HAS TO BE** a period ('.'), as in US format.

Calibration step 4 is now done.

– **Step 5: Voltage Drop**

Select "*Step 5*" tab in the calibration spreadsheet file.

Connect the amp-meter, the volt-meter and the power supply to the DC load, for current **AND** voltage measurements. Sets the output voltage to **5V** (the last entry in the array should be set around **12V**).

Using the DAC command, try to adjust the its value to match each value in the "***mA***" column on the amp-meter, and write down the voltage readed value on the volt-meter into the "***mV meter***" column, and the readed value on the LCD and/or serial terminal emulator to the "***mV LCD/Term.***" column.

The DAC command syntax is :

```
:DAC:value
```

where value is an integer from 0 to 4095.

You can change the values in the "***mA***" column to strictly match the ones you're reading on the ampmeter.

For the last row on the array, set the output voltage to around **12V**.

Once you went through the whole array, set DAC value to **0**:

```
:DAC:0
```

The calibration string should be entered into the serial terminal emulator, like:

```
:CAL:VD:x.xxx,y.yyy
```

Please note that the decimal separator **HAS TO BE** a period ('.'), as in US format.

Calibration step 5 is now done.

– **Last Step: Backup**

Once the full calibration is done, you **HAVE** to save the values into the EEPROM, using the following command:

```
:CAL:SAVE
```

Now, the calibration is done. You can use your DC load.

Chapter 8

Hierarchical Index

8.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 9

Class Index

9.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 10

File Index

10.1 File List

Here is a list of all files with brief descriptions:

Chapter 11

Class Documentation

11.1 aDCSettings::_eepromCalibrationValue_t Union Reference

Union to manipulate float/uint8_t [] calibration values.

Public Attributes

- float v
- uint8_t c [sizeof(float)]

11.1.1 Detailed Description

Union to manipulate float/uint8_t [] calibration values.

11.1.2 Member Data Documentation

11.1.2.1 uint8_t aDCSettings::_eepromCalibrationValue_t::c[sizeof(float)]

11.1.2.2 float aDCSettings::_eepromCalibrationValue_t::v

The documentation for this union was generated from the following file:

- aDCLoad.h

11.2 aDCDisplay Class Reference

Class that handles LCD displaying.

```
#include <aDCLoad.h>
```

Inherits aLCD.

Inherited by aDCEngine.

Public Member Functions

- aDCDisplay (aDCEngine *, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t)

  Constructor.

- ~aDCDisplay ()

  Destructor.
- void setup ()

  Setup function. Should be called before any other member.
- void showBanner ()

  Display small banner.
- void updateField (aDCSettings::OperationMode_t, float, float, uint8_t, uint8_t)

  Update displayed field according to operation mode.
- void updateDisplay ()

  Update LCD display management function.
- void pingBacklight ()

  Reset backlight dimmer.
- bool isBacklightDimmed ()

  Check if backlight is dimmed.

Private Member Functions

- void _dimBacklight (bool)

  Dim/undim backlight helper function.
- void _dimmingBacklight ()

  Backlight dimming.
- void _wakeupBacklight ()

  Backlight waker.

Private Attributes

- aDCEngine * m_Parent

  Pointer to aDCEngine parent.
- bool m_dimmed

  Dimmed state storage.
- unsigned long m_dimmerTick

  Dimmer timeout tick counter.
- unsigned long m_nextUpdate

11.2.1 Detailed Description

Class that handles LCD displaying.

11.2.2 Constructor & Destructor Documentation

11.2.2.1 aDCDisplay::aDCDisplay (aDCEngine * *parent*, uint8_t *rs*, uint8_t *enable*, uint8_t *d4*, uint8_t *d5*, uint8_t *d6*, uint8_t *d7*, uint8_t *cols*, uint8_t *rows*)

Constructor.

Class to manage display output

Parameters

parent	aDCEngine∗ : **Parent engine**
rs	uint8_t : **LCD RS pin**
enable	uint8_t : **LCD ENABLE pin**
d4	uint8_t : **LCD d4 pin**
d5	uint8_t : **LCD d5 pin**
d6	uint8_t : **LCD d6 pin**
d7	uint8_t : **LCD d7 pin**
cols	uint8_t : **LCD columns**
rows	uint8_t : **LCD rows**

11.2.2.2 aDCDisplay::∼aDCDisplay ( )

Destructor.

11.2.3 Member Function Documentation

11.2.3.1 void aDCDisplay::_dimBacklight (bool *up*) [private]

Dim/undim backlight helper function.

Parameters

up	bool : **dimmer direction**

Returns

void

11.2.3.2 void aDCDisplay::_dimmingBacklight ( ) [private]

Backlight dimming.

Returns

void

11.2.3.3 void aDCDisplay::_wakeupBacklight ( ) [private]

Backlight waker.

Returns

void

11.2.3.4 bool aDCDisplay::isBacklightDimmed ( )

Check if backlight is dimmed.

Returns

bool

11.2.3.5 void aDCDisplay::pingBacklight ( )

Reset backlight dimmer.

Returns

void

11.2.3.6 void aDCDisplay::setup ( )

Setup function. Should be called before any other member.

Returns

void

11.2.3.7 void aDCDisplay::showBanner ( )

Display small banner.

Returns

void

11.2.3.8 void aDCDisplay::updateDisplay ( )

Update LCD display management function.

Draw/Redraw on screen datas

Returns

void

11.2.3.9 void aDCDisplay::updateField (aDCSettings::OperationMode_t *opMode,* float *vSet,* float *vRead,* uint8_t *row,* uint8_t *unit*)

Update displayed field according to operation mode.

Parameters

opMode	OperationMode_t : **Operation mode**
vSet	float : **Settings value**
vRead	float : **Readed value**
row	uint8_t : **LCD row position**
unit	uint8_t : **Unit character**

Returns

void

11.2.4 Member Data Documentation

11.2.4.1 bool aDCDisplay::m_dimmed [private]

Dimmed state storage.

11.2.4.2 unsigned long aDCDisplay::m_dimmerTick [private]

Dimmer timeout tick counter.

11.2.4.3 unsigned long aDCDisplay::m_nextUpdate [private]

11.2.4.4 aDCEngine* aDCDisplay::m_Parent [private]

Pointer to aDCEngine parent.

The documentation for this class was generated from the following files:

- aDCLoad.h
- aDCLoad.cpp

11.3 aDCEngine Class Reference

Main class.

```
#include <aDCLoad.h>
```

Inherits aDCDisplay.

Public Member Functions

- aDCEngine (uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t=1)

  Main engine constructor.
- ~aDCEngine ()

  Destructor.
- void setup ()

  Setup function, should be called before any other member.
- void run ()

  Main loop function.
- const aDCSettings * getSettings () const

  Return a pointer to settings instantiated class.
- const ClickEncoder * getEncoder () const
- void setInput (bool)

Private Member Functions

- void _handleButtonEvent (ClickEncoder::Button)

  Function that handles button clicking.
- float _getInputVoltage ()

  Function to read the input voltage and return a float number represention volts.
- float _getADC (uint8_t)

  Function to read the ADC, accepts the channel to be read.
- void _setDAC (uint16_t, uint8_t)

  Function to set the DAC, Accepts the Value to be sent and the channel of the DAC to be used.
- int16_t _getTemp ()

  Function that read temperature from ADC channels.
- void _updateFanSpeed ()

Function to set the fan speed depending on the temperature sensors value.

- float _getMeasuredCurrent ()

  Function to measure the actual load current.

- void _updateLoadCurrent ()

  Function to calculate and set the required load current. Accepts the mode variable to determine if the constant current, resistance or power mode is to be used.

- void _adjustLoadCurrent ()

  Adjust current settings.

- void _handleLoggingAndRemote ()

  Check and handle remote control and data logging.

Private Attributes

- aDCSettings m_Data

  Settings object.

- ClickEncoder * m_encoder

  Encoder object.

- uint8_t m_RXbuffer [RXBUFFER_MAXLEN]

  USB rx buffer.

- uint8_t m_RXoffset

  USB rx buffer offset counter.

- aInputRelay m_inputRelay

Static Private Attributes

- static const uint8_t RXBUFFER_MAXLEN = 64

Friends

- class aDCDisplay

11.3.1 Detailed Description

Main class.

11.3.2 Constructor & Destructor Documentation

11.3.2.1 aDCEngine::aDCEngine (uint8_t *rs*, uint8_t *enable*, uint8_t *d4*, uint8_t *d5*, uint8_t *d6*, uint8_t *d7*, uint8_t *cols*, uint8_t *rows*, uint8_t *irbtn*, uint8_t *ir*, uint8_t *enca*, uint8_t *encb*, uint8_t *encpb*, uint8_t *encsteps* = `1`)

Main engine constructor.

Class to manage settings

Parameters

rs	uint8_t : **LCD RS pin**
enable	uint8_t : **LCD ENABLE pin**

d4	uint8_t : **LCD d4 pin**
d5	uint8_t : **LCD d5 pin**
d6	uint8_t : **LCD d6 pin**
d7	uint8_t : **LCD d7 pin**
cols	uint8_t : **LCD Columns number**
rows	uint8_t : **LCD Rows number**
irbtn	uint8_t : **Input Relay button pin**
ir	uint8_t : **Input Relay command pin**
enca	uint8_t : **Encoder A pin**
encb	uint8_t : **Encoder B pin**
encpb	uint8_t : **Encoder push button pin**
encsteps	uint8_t : **Encoder steps per notch**

11.3.2.2 aDCEngine::∼aDCEngine ( )

Destructor.

11.3.3 Member Function Documentation

11.3.3.1 void aDCEngine::_adjustLoadCurrent ( ) [private]

Adjust current settings.

Returns

void

11.3.3.2 float aDCEngine::_getADC (uint8_t *channel*) [private]

Function to read the ADC, accepts the channel to be read.

Parameters

channel	uint8_t : **ADC channel**

Returns

float : **readed value**

11.3.3.3 float aDCEngine::_getInputVoltage ( ) [private]

Function to read the input voltage and return a float number represention volts.

Returns

float : **readed input voltage**

11.3.3.4 float aDCEngine::_getMeasuredCurrent ( ) [private]

Function to measure the actual load current.

Returns

float : **readed current**

11.3.3.5 int16_t aDCEngine::_getTemp ( ) [private]

Function that read temperature from ADC channels.

Returns

int16_t : **temperature**

11.3.3.6 void aDCEngine::_handleButtonEvent (ClickEncoder::Button *button*) [private]

Function that handles button clicking.

Parameters

button	ClickEncoder::Button

Returns

void

11.3.3.7 void aDCEngine::_handleLoggingAndRemote ( ) [private]

Check and handle remote control and data logging.

Returns

void

11.3.3.8 void aDCEngine::_setDAC (uint16_t *value,* uint8_t *channel*) [private]

Function to set the DAC, Accepts the Value to be sent and the channel of the DAC to be used.

Parameters

value	uint16_t : **value to send to DAC**
channel	uint8_t : **DAC channel**

Returns

void

11.3.3.9 void aDCEngine::_updateFanSpeed ( ) [private]

Function to set the fan speed depending on the temperature sensors value.

Returns

void

11.3.3.10 void aDCEngine::_updateLoadCurrent ( ) [private]

Function to calculate and set the required load current. Accepts the mode variable to determine if the constant current, resistance or power mode is to be used.

Returns

void

11.3.3.11 const ClickEncoder * aDCEngine::getEncoder ( ) const

11.3.3.12 const aDCSettings * aDCEngine::getSettings ( ) const

Return a pointer to settings instantiated class.

Returns

const aDCSettings*

11.3.3.13 void aDCEngine::run ( )

Main loop function.

Returns

void

11.3.3.14 void aDCEngine::setInput (bool *enable*)

Parameters

enable	bool

Returns

void

11.3.3.15 void aDCEngine::setup ( )

Setup function, should be called before any other member.

Returns

void

11.3.4 Friends And Related Function Documentation

11.3.4.1 friend class aDCDisplay [friend]

11.3.5 Member Data Documentation

11.3.5.1 aDCSettings aDCEngine::m_Data [private]

Settings object.

11.3.5.2 ClickEncoder* aDCEngine::m_encoder [private]

Encoder object.

11.3.5.3 aInputRelay aDCEngine::m_inputRelay [private]

11.3.5.4 uint8_t aDCEngine::m_RXbuffer[RXBUFFER_MAXLEN] [private]

USB rx buffer.

11.3.5.5 uint8_t aDCEngine::m_RXoffset [private]

USB rx buffer offset counter.

11.3.5.6 const uint8_t aDCEngine::RXBUFFER_MAXLEN = 64 [static], [private]

The documentation for this class was generated from the following files:

- aDCLoad.h
- aDCLoad.cpp

11.4 aDCSettings Class Reference

Class that handles settings.

```
#include <aDCLoad.h>
```

Inherits aStepper.

Classes

- union _eepromCalibrationValue_t

  Union to manipulate float/uint8_t [] calibration values.
- struct CalibrationData_t

  Calibration values.

Public Types

- enum OperationMode_t { OPERATION_MODE_READ, OPERATION_MODE_SET, OPERATION_MODE_UNKNOWN }

  Operation mode enumeration.
- enum SelectionMode_t { SELECTION_MODE_CURRENT, SELECTION_MODE_POWER, SELECTION_MODE_PULSE, SELECTION_MODE_UNKNOWN }

  Selection settings mode enumeration.
- enum DisplayMode_t { DISPLAY_MODE_VALUES, DISPLAY_MODE_SETUP, DISPLAY_MODE_UNKNOWN }

  Display mode enumeration.
- enum SettingError_t { SETTING_ERROR_OVERSIZED, SETTING_ERROR_UNDERSIZED, SETTING_ERROR_VALID }

  Setting error enumeration.
- enum CalibrationValues_t { CALIBRATION_VOLTAGE, CALIBRATION_READ_CURRENT, CALIBRATION_DAC_CURRENT, CALIBRATION_VOLTAGE_DROP, CALIBRATION_MAX }

  Calibration offset enumeration.

Public Member Functions

- aDCSettings (aDCEngine *)

  Constructor.
- ~aDCSettings ()

  Destructor.

- SettingError_t setVoltage (float)

  Voltage setter.
- float getVoltage ()

  Voltage getter.
- SettingError_t setCurrent (float, OperationMode_t)

  Current setter.
- float getCurrent (OperationMode_t)

  Current getter.
- SettingError_t setPulse (float)

  Pulse setter.
- float getPulse ()

  Pulse getter.
- bool isPulseEnabled ()

  Returns pulse enability.
- void enablePulse (bool)

  Set pulse enability.
- bool isPulseHigh ()

  Returns if pulse is high (load should operate)
- void setPulseHigh (bool)

  Set pulse high/low state value.
- SettingError_t setPower (float, OperationMode_t)

  Power setter.
- float getPower (OperationMode_t)

  Power getter.
- void updateValuesFromMode (float, SelectionMode_t)

  Update values setting (Current, Resistance, Power) according to selection mode. Sanity checking is also performed.
- void setTemperature (uint16_t)

  Temperature readed setter.
- uint16_t getTemperature ()

  Temperature readed getter.
- void setFanSpeed (uint16_t)

  Fan speed setter.
- uint16_t getFanSpeed ()

  Fan speed getter.
- void setCurrentDAC (uint16_t)

  Current DAC value setter.
- uint16_t getCurrentDAC ()

  Current DAC value getter.
- void setSelectionMode (SelectionMode_t, bool=false)

  Selection mode setter.
- SelectionMode_t getSelectionMode ()

  Selection mode getter.
- SelectionMode_t getPrevNextMode (SelectionMode_t=SELECTION_MODE_UNKNOWN, bool=true)

  Get the next selection mode, according to "origin", if any provided.
- void setDisplayMode (DisplayMode_t)

  Display mode setter.
- DisplayMode_t getDisplayMode ()

  Display mode getter.
- void setEncoderPosition (int32_t)

  Encoder position setter.
- void incEncoderPosition (int32_t=1)

Increment stored encoder position by "p" (default = 1)

- int32_t getEncoderPosition ()

  Encoder position getter.
- void setOperationMode (OperationMode_t)

  Operation mode setter.
- OperationMode_t getOperationMode ()

  Operation mode getter.
- void updateOperationMode ()

  Automatic timeouted toggle between OPERATION_SET and OPERATION_READ.
- void pingOperationMode ()

  Reset timeout while in OPERATION_SET mode.
- void pingAutolock ()

  Reset autolock timeout.
- bool isAutolocked ()

  Check if autolock is enabled and performs.
- void setCalibationMode (bool=true)

  Calibration mode enability setter.
- bool getCalibrationMode ()

  Calibration mode enability getter.
- void getCalibrationValues (CalibrationValues_t, CalibrationData_t &)

  Calibration data getter, according to calsection argument.
- void setCalibrationValues (CalibrationValues_t, CalibrationData_t)

  Calibration data setter, according to calsection argument.
- void backupCalibration ()

  Backup calibration data into EEPROM.
- void restoreCalibration ()

  Restore calibration data from EEPROM.
- void enableAlarm (uint16_t bit)

  Turn alarm (OVP, OCP or OTP) on, sets output current to zero.
- void enableFeature (uint16_t, bool=true)

  Helper function to manage bit-field features.
- bool isFeatureEnabled (uint16_t)

  Check if FEATURE is enabled.
- bool isDataEnabled (uint16_t)

  Get bit enability in m_datas bit-field storage.
- void syncData (uint16_t)

  Clear a bit in m_datas bit-field storage.

Static Public Attributes

- static const uint16_t DATA_VOLTAGE = 1

  bit-field storage: Voltage readed
- static const uint16_t DATA_CURRENT_SETS = 1 << 1

  bit-field storage: Current sets
- static const uint16_t DATA_CURRENT_READ = 1 << 2

  bit-field storage: Current readed
- static const uint16_t DATA_PULSE_SETS = 1 << 3
- static const uint16_t DATA_POWER_SETS = 1 << 5

  bit-field storage: Power sets
- static const uint16_t DATA_POWER_READ = 1 << 6

bit-field storage: Power readed

- static const uint16_t DATA_TEMPERATURE = 1 << 7

  bit-field storage: Temperature readed
- static const uint16_t DATA_SELECTION = 1 << 8

  bit-field storage: Selection mode sets
- static const uint16_t DATA_DISPLAY = 1 << 9

  bit-field storage: Display mode sets
- static const uint16_t DATA_ENCODER = 1 << 10

  bit-field storage: Encoder position sets
- static const uint16_t DATA_OPERATION = 1 << 11

  bit-field storage: Operation mode sets
- static const uint16_t DATA_IN_CALIBRATION = 1 << 12

  bit-field storage: In calibration mode

Private Member Functions

- SettingError_t _setValue (OperationMode_t, uint16_t, float, float &, float &, float)

  Value setter.
- uint8_t _crc8 (const uint8_t *, uint8_t)

  CRC8 computation.
- void _eepromCalibrationRestore (int16_t, CalibrationData_t &)

  Read data from EEPROM, used to restore calibration data.
- void _eepromCalibrationBackup (int16_t, CalibrationData_t)

  Save data to EEPROM, used to restore calibration data.
- bool _eepromCheckForMagicNumbers ()

  Check for EEPROM magic numbers.
- void _eepromWriteMagicNumbers ()

  Write magic numbers into EEPROM.
- void _eepromReset ()

  Reset all stored parameters into EEPROM.
- void _eepromRestore ()

  Restore value from EEPROM.
- void _enableData (uint16_t, bool)

  Enable a bit in the m_datas bit-field storage.
- void _enableDataCheck (uint16_t, bool)

  Enable a bit in the m_datas bit-field storage, checking for previous state.

Private Attributes

- aDCEngine * m_Parent
- float m_readVoltage

  voltage storage
- float m_setsCurrent
- float m_readCurrent

  current storage
- float m_setsPulse

  Pulse time length (ms)
- float m_setsPower
- float m_readPower

  power storage

- uint16_t m_readTemperature

  temperature storage
- uint16_t m_fanSpeed

  fan speed storage
- uint16_t m_currentDAC

  current DAC value
- OperationMode_t m_operationMode

  operation mode
- SelectionMode_t m_mode

  selection mode
- int32_t m_encoderPos

  encoder position (yeah, 32bits, due to resistance max val (12000 ∗ 1000);
- DisplayMode_t m_dispMode

  display mode
- unsigned long m_lockTick

  tick count storage, for autolock feature
- unsigned long m_operationTick

  tick count storage, for SET/READ operation
- uint16_t m_features

  boolean features storage
- uint16_t m_datas

  boolean displayes data storage
- CalibrationData_t m_calibrationValues [CALIBRATION_MAX]

  Calibration datas, restored from EEPROM.
- volatile bool m_pulseEnabled

  Pulse is enabled.
- volatile bool m_pulseHigh

  Pulse is high ?

11.4.1 Detailed Description

Class that handles settings.

11.4.2 Member Enumeration Documentation

11.4.2.1 enum aDCSettings::CalibrationValues_t

Calibration offset enumeration.

Used to get/set calibration values

Enumerator

CALIBRATION_VOLTAGE Voltage calibration offset value.

CALIBRATION_READ_CURRENT Readed Current calibration offset value.

CALIBRATION_DAC_CURRENT DAC Current calibration offset value.

CALIBRATION_VOLTAGE_DROP Voltage drop calibration offset value.

CALIBRATION_MAX Maximum offset in calibration array.

11.4.2.2 enum aDCSettings::DisplayMode_t

Display mode enumeration.

Enumerator

DISPLAY_MODE_VALUES Display read/set values mode.

DISPLAY_MODE_SETUP Display settings mode.

DISPLAY_MODE_UNKNOWN Display in undefined (internal)

11.4.2.3 enum aDCSettings::OperationMode_t

Operation mode enumeration.

Enumerator

OPERATION_MODE_READ Reading values.

OPERATION_MODE_SET Settings values.

OPERATION_MODE_UNKNOWN Unset (internal)

11.4.2.4 enum aDCSettings::SelectionMode_t

Selection settings mode enumeration.

Enumerator

SELECTION_MODE_CURRENT Current selected.

SELECTION_MODE_POWER Power selected.

SELECTION_MODE_PULSE Pulse selected.

SELECTION_MODE_UNKNOWN Nothing selected (internal)

11.4.2.5 enum aDCSettings::SettingError_t

Setting error enumeration.

Enumerator

SETTING_ERROR_OVERSIZED Setting value is oversized.

SETTING_ERROR_UNDERSIZED Setting value is undersized.

SETTING_ERROR_VALID Setting value is valid.

11.4.3 Constructor & Destructor Documentation

11.4.3.1 aDCSettings::aDCSettings (aDCEngine * *parent*)

Constructor.

Parameters

parent	aDCEngine∗ : **Pointer to aDCEngine parent**

11.4.3.2 aDCSettings::∼aDCSettings ( )

Destructor.

11.4.4 Member Function Documentation

11.4.4.1 uint8_t aDCSettings::_crc8 (const uint8_t ∗ *addr*, uint8_t *len*) [private]

CRC8 computation.

Code took from http://www.pjrc.com/teensy/td_libs_OneWire.html

Parameters

addr	const uint8_t∗ : **Data source**
len	uint8_t : **Data source length**

Returns

uint8_t : **CRC**

11.4.4.2 void aDCSettings::_eepromCalibrationBackup (int16_t *addr*, CalibrationData_t *cal*) [private]

Save data to EEPROM, used to restore calibration data.

Parameters

addr	int16_t : **start address location**
cal	CalibrationData_t& : **destination**

Returns

void

11.4.4.3 void aDCSettings::_eepromCalibrationRestore (int16_t *addr*, CalibrationData_t & *cal*) [private]

Read data from EEPROM, used to restore calibration data.

Parameters

addr	int16_t : **start address location**
cal	CalibrationData_t& : **destination**

Returns

void

11.4.4.4 bool aDCSettings::_eepromCheckForMagicNumbers ( ) [private]

Check for EEPROM magic numbers.

Returns

bool

Used to check if some data has already been wrote in the EEPROM.

11.4.4.5 void aDCSettings::_eepromReset ( ) [private]

Reset all stored parameters into EEPROM.

Returns

void

11.4.4.6 void aDCSettings::_eepromRestore ( ) [private]

Restore value from EEPROM.

Returns

void

11.4.4.7 void aDCSettings::_eepromWriteMagicNumbers ( ) [private]

Write magic numbers into EEPROM.

Returns

void

11.4.4.8 void aDCSettings::_enableData (uint16_t *bit*, bool *enable*) [private]

Enable a bit in the m_datas bit-field storage.

Parameters

bit	uint16_t : **Bit to set**
enable	bool : **Bit enability**

Returns

void

11.4.4.9 void aDCSettings::_enableDataCheck (uint16_t *bit*, bool *enable*) [private]

Enable a bit in the m_datas bit-field storage, checking for previous state.

If the bit is already sets to TRUE, we don't touch his state, syncData() should be called for this.

Parameters

bit	uint16_t : **Bit to set**
enable	bool : **Bit enability**

Returns

void

11.4.4.10 aDCSettings::SettingError_t aDCSettings::_setValue (OperationMode_t *mode*, uint16_t *bit*, float *value*, float & *sets*, float & *read*, float *maximum*) [private]

Value setter.

Parameters

mode	OperationMode_t : **Operation mode (SET/READ)**
bit	uint16_t : **DATA_∗ bit to set**
value	float : **value to store**
sets	float& : **destination variable for OPERATION_SET**
read	float& : **destination variable for OPERTION_READ**
maximum	float : **maximum value, used for boundaries checking**

Returns

SettingError_t : **validity result**

11.4.4.11 void aDCSettings::backupCalibration ( )

Backup calibration data into EEPROM.

Returns

void

11.4.4.12 void aDCSettings::enableAlarm (uint16_t *aBit*)

Turn alarm (OVP, OCP or OTP) on, sets output current to zero.

Parameters

aBit	uint16_t : **alarm bit to set**

Returns

void

11.4.4.13 void aDCSettings::enableFeature (uint16_t *feature*, bool *enable* = true)

Helper function to manage bit-field features.

Parameters

feature	uint16_t : **FEATURE to enable/disable**
enable	bool : **FEATURE enability (default = enable)**

Returns

void

11.4.4.14 void aDCSettings::enablePulse (bool *enable*)

Set pulse enability.

Parameters

enable	bool : **enability**

Returns

void

11.4.4.15 bool aDCSettings::getCalibrationMode ( )

Calibration mode enability getter.

Returns

bool : **enability**

11.4.4.16 void aDCSettings::getCalibrationValues (CalibrationValues_t *calsection,* CalibrationData_t & *data*)

Calibration data getter, according to *calsection* argument.

Parameters

calsection	CalibrationValues_t : **calibration parameter**
data	CalibrationData_t& : **calibration data**

Returns

void

11.4.4.17 float aDCSettings::getCurrent (OperationMode_t *mode*)

Current getter.

Parameters

mode	OperationMode_t : **READ/SET mode storage access**

Returns

float : **current**

11.4.4.18 uint16_t aDCSettings::getCurrentDAC ( )

Current DAC value getter.

Returns

uint16_t : **current DAC value**

11.4.4.19 aDCSettings::DisplayMode_t aDCSettings::getDisplayMode ( )

Display mode getter.

Returns

DisplayMode_t : **Display mode**

11.4.4.20 int32_t aDCSettings::getEncoderPosition ( )

Encoder position getter.

Returns

int32_t : **encoder position**

11.4.4.21 uint16_t aDCSettings::getFanSpeed ( )

Fan speed getter.

Returns

uint16_t : **DAC speed value**

11.4.4.22 aDCSettings::OperationMode_t aDCSettings::getOperationMode ( )

Operation mode getter.

Returns

OperationMode_t : **operation mode**

11.4.4.23 float aDCSettings::getPower (OperationMode_t *mode*)

Power getter.

Parameters

mode	OperationMode_t : **READ/SET mode storage access**

Returns

float : **power**

11.4.4.24 aDCSettings::SelectionMode_t aDCSettings::getPrevNextMode (aDCSettings::SelectionMode_t *origin* = SELECTION_MODE_UNKNOWN, bool *next* = `true`)

Get the next selection mode, according to "origin", if any provided.

Parameters

origin	SelectionMode_t : **origin starter selection mode**
next	bool : **Next or Previous mode**

Returns

SelectionMode_t : **next selection mode**

11.4.4.25 float aDCSettings::getPulse ( )

Pulse getter.

Returns

float : **pulse (ms)**

11.4.4.26 aDCSettings::SelectionMode_t aDCSettings::getSelectionMode ( )

Selection mode getter.

Returns

SelectionMode_t : **current selection mode**

11.4.4.27 uint16_t aDCSettings::getTemperature ( )

Temperature readed getter.

Returns

uint16_t : **temperature**

11.4.4.28 float aDCSettings::getVoltage ( )

Voltage getter.

Returns

float : **voltage**

11.4.4.29 void aDCSettings::incEncoderPosition (int32_t *v* = `1`)

Increment stored encoder position by "p" (default = 1)

Parameters

v	int32_t : **increment value, 1 by default**

Returns

void

11.4.4.30 bool aDCSettings::isAutolocked ( )

Check if autolock is enabled and performs.

Returns

bool

11.4.4.31 bool aDCSettings::isDataEnabled (uint16_t *bit*)

Get bit enability in m_datas bit-field storage.

Parameters

bit	uint16_t : **Bit to check**

Returns

bool

11.4.4.32 bool aDCSettings::isFeatureEnabled (uint16_t *feature*)

Check if FEATURE is enabled.

Parameters

feature	uint16_t : **FEATURE to check against**

Returns

bool

11.4.4.33 bool aDCSettings::isPulseEnabled ( )

Returns pulse enability.

Returns

bool

11.4.4.34 bool aDCSettings::isPulseHigh ( )

Returns if pulse is high (load should operate)

Returns

bool

11.4.4.35 void aDCSettings::pingAutolock ( )

Reset autolock timeout.

Returns

void

11.4.4.36 void aDCSettings::pingOperationMode ( )

Reset timeout while in OPERATION_SET mode.

Returns

void

11.4.4.37 void aDCSettings::restoreCalibration ( )

Restore calibration data from EEPROM.

Returns

void

11.4.4.38 void aDCSettings::setCalibationMode (bool *enable* = `true`)

Calibration mode enability setter.

Parameters

enable	bool : **enability**

Returns

void

11.4.4.39 void aDCSettings::setCalibrationValues (CalibrationValues_t *calsection*, CalibrationData_t *data*)

Calibration data setter, according to *calsection* argument.

Parameters

calsection	CalibrationValues_t : **calibration parameter**
data	CalibrationData_t& : **calibration data**

Returns

void

11.4.4.40 aDCSettings::SettingError_t aDCSettings::setCurrent (float *v*, OperationMode_t *mode*)

Current setter.

Parameters

v	float : **current**
mode	OperationMode_t : **READ/SET mode storage access**

Returns

aDCSettings::SettingError_t

11.4.4.41 void aDCSettings::setCurrentDAC (uint16_t *dac*)

Current DAC value setter.

Parameters

dac	uint16_t : **current DAC value**

Returns

void

11.4.4.42 void aDCSettings::setDisplayMode (DisplayMode_t *d*)

Display mode setter.

Parameters

d	DisplayMode_t : **display mode**

Returns

void

11.4.4.43 void aDCSettings::setEncoderPosition (int32_t *v*)

Encoder position setter.

Parameters

v	int32_t : **encoder position**

Returns

void

11.4.4.44 void aDCSettings::setFanSpeed (uint16_t *v*)

Fan speed setter.

Parameters

v	uint16_t : **DAC speed value**

Returns

void

11.4.4.45 void aDCSettings::setOperationMode (OperationMode_t *m*)

Operation mode setter.

Parameters

m	OperationMode_t : **new operation mode**

Returns

void

11.4.4.46 aDCSettings::SettingError_t aDCSettings::setPower (float *v*, OperationMode_t *mode*)

Power setter.

Parameters

v	float : **power**
mode	OperationMode_t : **READ/SET mode storage access**

Returns

aDCSettings::SettingError_t

11.4.4.47 aDCSettings::SettingError_t aDCSettings::setPulse (float *v*)

Pulse setter.

Parameters

v	float : **pulse (ms)**

Returns

aDCSettings::SettingError_t

< Sleep well honey

< Wake up son!

11.4.4.48 void aDCSettings::setPulseHigh (bool *high*)

Set pulse high/low state value.

Parameters

high	bool : **pulse state**

Returns

void

11.4.4.49 void aDCSettings::setSelectionMode (SelectionMode_t *m*, bool *force* = `false`)

Selection mode setter.

Parameters

m	SelectionMode_t : **new selection mode**
force	bool : **force the bit setting in m_datas to be set (default : false)**

Returns

void

11.4.4.50 void aDCSettings::setTemperature (uint16_t *v*)

Temperature readed setter.

Parameters

v	uint16_t : **temperature**

Returns

void

11.4.4.51 aDCSettings::SettingError_t aDCSettings::setVoltage (float *v*)

Voltage setter.

Parameters

v	float : **voltage**

Returns

aDCSettings::SettingError_t

11.4.4.52 void aDCSettings::syncData (uint16_t *bit*)

Clear a bit in m_datas bit-field storage.

Parameters

bit	uint16_t

Returns

void

11.4.4.53 void aDCSettings::updateOperationMode ( )

Automatic timeouted toggle between OPERATION_SET and OPERATION_READ.

Returns

void

11.4.4.54 void aDCSettings::updateValuesFromMode (float *v*, SelectionMode_t *mode*)

Update values setting (Current, Resistance, Power) according to selection mode. Sanity checking is also performed.

Parameters

v	float : **updated value**
mode	SelectionMode_t : **selection mode (CURRENT, RESISTANCE, POWER)**

Returns

void

11.4.5 Member Data Documentation

11.4.5.1 const uint16_t aDCSettings::DATA_CURRENT_READ = 1 << 2 [static]

bit-field storage: Current readed

11.4.5.2 const uint16_t aDCSettings::DATA_CURRENT_SETS = 1 << 1 [static]

bit-field storage: Current sets

11.4.5.3 const uint16_t aDCSettings::DATA_DISPLAY = 1 << 9 [static]

bit-field storage: Display mode sets

11.4.5.4 const uint16_t aDCSettings::DATA_ENCODER = 1 << 10 [static]

bit-field storage: Encoder position sets

11.4.5.5 const uint16_t aDCSettings::DATA_IN_CALIBRATION = 1 << 12 [static]

bit-field storage: In calibration mode

11.4.5.6 const uint16_t aDCSettings::DATA_OPERATION = 1 << 11 [static]

bit-field storage: Operation mode sets

11.4.5.7 const uint16_t aDCSettings::DATA_POWER_READ = 1 << 6 [static]

bit-field storage: Power readed

11.4.5.8 const uint16_t aDCSettings::DATA_POWER_SETS = 1 << 5 [static]

bit-field storage: Power sets

11.4.5.9 const uint16_t aDCSettings::DATA_PULSE_SETS = 1 << 3 [static]

11.4.5.10 const uint16_t aDCSettings::DATA_SELECTION = 1 << 8 [static]

bit-field storage: Selection mode sets

11.4.5.11 const uint16_t aDCSettings::DATA_TEMPERATURE = 1 << 7 [static]

bit-field storage: Temperature readed

11.4.5.12 const uint16_t aDCSettings::DATA_VOLTAGE = 1 [static]

bit-field storage: Voltage readed

11.4.5.13 CalibrationData_t aDCSettings::m_calibrationValues[CALIBRATION_MAX] [private]

Calibration datas, restored from EEPROM.

11.4.5.14 uint16_t aDCSettings::m_currentDAC [private]

current DAC value

11.4.5.15 uint16_t aDCSettings::m_datas [private]

boolean displayes data storage

11.4.5.16 DisplayMode_t aDCSettings::m_dispMode [private]

display mode

See Also

DisplayMode

11.4.5.17 int32_t aDCSettings::m_encoderPos [private]

encoder position (yeah, 32bits, due to resistance max val (12000 $*$ 1000);

11.4.5.18 uint16_t aDCSettings::m_fanSpeed [private]

fan speed storage

11.4.5.19 uint16_t aDCSettings::m_features [private]

boolean features storage

11.4.5.20 unsigned long aDCSettings::m_lockTick [private]

tick count storage, for autolock feature

11.4.5.21 SelectionMode_t aDCSettings::m_mode [private]

selection mode

See Also

SelectionMode

11.4.5.22 OperationMode_t aDCSettings::m_operationMode [private]

operation mode

See Also

OperationMode

11.4.5.23 unsigned long aDCSettings::m_operationTick [private]

tick count storage, for SET/READ operation

11.4.5.24 aDCEngine* aDCSettings::m_Parent [private]

11.4.5.25 volatile bool aDCSettings::m_pulseEnabled [private]

Pulse is enabled.

11.4.5.26 volatile bool aDCSettings::m_pulseHigh [private]

Pulse is high ?

11.4.5.27 float aDCSettings::m_readCurrent [private]

current storage

11.4.5.28 float aDCSettings::m_readPower [private]

power storage

11.4.5.29 uint16_t aDCSettings::m_readTemperature [private]

temperature storage

11.4.5.30 float aDCSettings::m_readVoltage [private]

voltage storage

11.4.5.31 float aDCSettings::m_setsCurrent [private]

11.4.5.32 float aDCSettings::m_setsPower [private]

11.4.5.33 float aDCSettings::m_setsPulse [private]

Pulse time length (ms)

The documentation for this class was generated from the following files:

- aDCLoad.h
- aDCLoad.cpp

11.5 aInputRelay Class Reference

Class that handles input relay.

```
#include <aDCLoad.h>
```

Public Member Functions

- aInputRelay (uint8_t, uint8_t)

  Constructor.
- ~aInputRelay ()

  Destructor.
- void service ()

  Function to check button press checking, should be called inside the main loop.
- bool isInput ()

  Returns the relay's state.
- void setInput (bool=true)

  Set relay's input state.
- bool isClicked ()

  Returns true if button has been clicked (pressed then released)

Private Attributes

- uint8_t m_btnPin
- uint8_t m_relayPin
- uint8_t m_btnState
- bool m_clicked
- bool m_isON

11.5.1 Detailed Description

Class that handles input relay.

11.5.2 Constructor & Destructor Documentation

11.5.2.1 aInputRelay::aInputRelay (uint8_t *btnPin*, uint8_t *relayPin*)

Constructor.

Class to manage input relay

Parameters

btnPin	uint8_t : **Button pin**
relayPin	uint8_t : **Relay pin**

11.5.2.2 aInputRelay::~aInputRelay ( )

Destructor.

11.5.3 Member Function Documentation

11.5.3.1 bool aInputRelay::isClicked ( )

Returns true if button has been clicked (pressed then released)

Returns

bool

11.5.3.2 bool aInputRelay::isInput ( )

Returns the relay's state.

Returns

bool

11.5.3.3 void aInputRelay::service ( )

Function to check button press checking, should be called inside the main loop.

Returns

void

11.5.3.4 void aInputRelay::setInput (bool *on* = `true`)

Set relay's input state.

Parameters

on	bool

Returns

void

11.5.4 Member Data Documentation

11.5.4.1 uint8_t aInputRelay::m_btnPin [private]

11.5.4.2 uint8_t aInputRelay::m_btnState [private]

11.5.4.3 bool aInputRelay::m_clicked [private]

11.5.4.4 bool aInputRelay::m_isON [private]

11.5.4.5 uint8_t aInputRelay::m_relayPin [private]

The documentation for this class was generated from the following files:

- aDCLoad.h
- aDCLoad.cpp

11.6 aLCD Class Reference

LiquidDisplay extension class.

`#include <aDCLoad.h>`

Inherits LiquidCrystal.

Inherited by aDCDisplay.

Public Member Functions

- aLCD (uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t)

  Constructor.
- ~aLCD ()

  Destructor.
- void begin (uint8_t, uint8_t)

  Initialize LCD screen size.
- void setCursor (uint8_t, uint8_t)

  Set cursor positon.
- void printCenter (const char *)

  Print centered string to current row.
- void printCenter (const __FlashStringHelper *)

  Print centered string to current row.
- void clearValue (uint8_t, int=0)

  Clear displayed value, from given row, stopping at value end field - destMinus.

Private Attributes

- uint8_t m_cols
- uint8_t m_rows

  LCD sizes.
- uint8_t m_curCol
- uint8_t m_curRow

  Current cursor position.

11.6.1 Detailed Description

LiquidDisplay extension class.

11.6.2 Constructor & Destructor Documentation

11.6.2.1 aLCD::aLCD (uint8_t *rs*, uint8_t *enable*, uint8_t *d4*, uint8_t *d5*, uint8_t *d6*, uint8_t *d7*, uint8_t *cols*, uint8_t *rows*)

Constructor.

LiquidDisplay extension

Parameters

rs	uint8_t : **LCD RS pin**
enable	uint8_t : **LCD ENABLE pin**
d4	uint8_t : **LCD d4 pin**
d5	uint8_t : **LCD d5 pin**
d6	uint8_t : **LCD d6 pin**
d7	uint8_t : **LCD d7 pin**
cols	uint8_t : **LCD columns number**
rows	uint8_t : **LCD rows number**

11.6.2.2 aLCD::~aLCD ( )

Destructor.

11.6.3 Member Function Documentation

11.6.3.1 void aLCD::begin (uint8_t *cols*, uint8_t *lines*)

Initialize LCD screen size.

Parameters

cols	uint8_t : **Columns**
lines	uint8_t : **Rows**

Returns

void

11.6.3.2 void aLCD::clearValue (uint8_t *row*, int *destMinus* = 0)

Clear displayed value, from given row, stopping at value end field - destMinus.

Parameters

row	uint8_t : **field row**
destMinus	int : **minus end field position (default = 0)**

Returns

void

11.6.3.3 void aLCD::printCenter (const char * *str*)

Print centered string to current row.

Parameters

str	const char* : **String to display**

Returns

void

11.6.3.4 void aLCD::printCenter (const __FlashStringHelper * *ifsh*)

Print centered string to current row.

Parameters

ifsh	const __FlashStringHelper* : **string to display**

Returns

void

11.6.3.5 void aLCD::setCursor (uint8_t *col*, uint8_t *row*)

Set cursor positon.

Parameters

col	uint8_t : **Column**
row	uint8_t : **Row**

Returns

void

11.6.4 Member Data Documentation

11.6.4.1 uint8_t aLCD::m_cols [private]

11.6.4.2 uint8_t aLCD::m_curCol [private]

11.6.4.3 uint8_t aLCD::m_curRow [private]

Current cursor position.

11.6.4.4 uint8_t aLCD::m_rows [private]

LCD sizes.

The documentation for this class was generated from the following files:

- aDCLoad.h
- aDCLoad.cpp

11.7 aStepper Class Reference

Class that handles increase/decrease step multiplier.

```
#include <aDCLoad.h>
```

Inherited by aDCSettings.

Public Member Functions

- aStepper ()

  Constructor.
- ~aStepper ()

  Destructor.
- void increment ()

  Increment value, check for boundaries.
- uint8_t getValue ()

  Value getter.
- void reset ()

  Reset value.
- int16_t getMult ()

  Value getter, according to multiple.
- int16_t getValueFromMode (uint8_t)

  Get value according to selection mode.
- bool isSynced ()

Synchronize value.

- void sync ()

  Check if value is synchronized.

Private Member Functions

- int16_t _pow (int, int)

  Small implementation of pow() math function.

Private Attributes

- uint8_t m_inc
- uint8_t m_incPrev

  Stepper counters.

Static Private Attributes

- static const uint8_t MAX_VALUE = 3

11.7.1 Detailed Description

Class that handles increase/decrease step multiplier.

Classes declarations

11.7.2 Constructor & Destructor Documentation

11.7.2.1 aStepper::aStepper ( )

Constructor.

11.7.2.2 aStepper::~aStepper ( )

Destructor.

11.7.3 Member Function Documentation

11.7.3.1 int16_t aStepper::_pow (int *base*, int *exp*) [inline], [private]

Small implementation of pow() math function.

Parameters

base	int : **base radix**
exp	int : **exponent value**

Returns

int16_t : **result**

11.7.3.2 int16_t aStepper::getMult ( )

Value getter, according to multiple.

Returns

int16_t

11.7.3.3 uint8_t aStepper::getValue ( )

Value getter.

Returns

uint8_t

11.7.3.4 int16_t aStepper::getValueFromMode (uint8_t *mode*)

Get value according to selection mode.

Parameters

mode	uint16_t : **Selection mode (will be typecasted to aDCSettings::SelectionMode_t)**

Returns

int16_t

11.7.3.5 void aStepper::increment ( )

Increment value, check for boundaries.

Returns

void

11.7.3.6 bool aStepper::isSynced ( )

Synchronize value.

Returns

bool

11.7.3.7 void aStepper::reset ( )

Reset value.

Returns

void

11.7.3.8 void aStepper::sync ( )

Check if value is synchronized.

Returns

void

11.7.4 Member Data Documentation

11.7.4.1 uint8_t aStepper::m_inc [private]

11.7.4.2 uint8_t aStepper::m_incPrev [private]

Stepper counters.

11.7.4.3 const uint8_t aStepper::MAX_VALUE = 3 [static],[private]

The documentation for this class was generated from the following files:

- aDCLoad.h
- aDCLoad.cpp

11.8 aDCSettings::CalibrationData_t Struct Reference

Calibration values.

```
#include <aDCLoad.h>
```

Public Attributes

- float slope

  Slope value.
- float offset

  Offset value.

11.8.1 Detailed Description

Calibration values.

Contains Slope and Offset float values

11.8.2 Member Data Documentation

11.8.2.1 float aDCSettings::CalibrationData_t::offset

Offset value.

11.8.2.2 float aDCSettings::CalibrationData_t::slope

Slope value.

The documentation for this struct was generated from the following file:

- aDCLoad.h

Chapter 12

File Documentation

12.1 aDCLoad.cpp File Reference

```
#include <Arduino.h>
#include <avr/pgmspace.h>
#include <LiquidCrystal.h>
#include <SPI.h>
#include <EEPROM.h>
#include <ClickEncoder.h>
#include <TimerOne.h>
#include <TimerThree.h>
#include "aDCLoad.h"
```

Functions

- void serialPrint (unsigned long n, int base)

  Serial printing.
- void serialWrite (char c)

  Serial printing.
- void serialPrint (const char str[])

  Serial printing.
- void serialPrint (char c)

  Serial printing.
- void serialPrint (int n, int16_t base=DEC)

  Serial printing.
- void serialFlush ()

  Flush serial buffer (TX)
- void serialPrintln ()

  Serial printing.
- void serialPrint (double n, int digits)

  Serial printing.
- void serialPrintln (double n, int digits)

  Serial printing.
- void serialPrintln (char c)

  Serial printing.
- int16_t serialAvailable ()

  Get the number of bytes (characters) available for reading from the serial port.
- uint8_t serialRead ()

Reads incoming serial data.

- int8_t getNumericalLength (float n, uint8_t len=3)

  Get float string length, according to decimal man length.
- float floatRounding (float f)

  Float number rounding, extensively used.
- static void timer1ISR (void)

  ISR callback function for Timer1, used for encoder.
- static void timer3ISR (void)

  ISR callback function for Timer3, used for pulse feature.

Variables

- aDCEngine * pThis = NULL

12.1.1 Detailed Description

Copyright

Copyright (C) 2014-2015 F1RMB, Daniel Caujolle-Bert `f1rmb.daniel@gmail.com`
Copyright (C) 2014 Lee Wiggins `lee@wigweb.com.au`

License:

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Warning

BIG FAT WARNING

Should be compiled with **"-Os"** flag.

Bootloader **couldn't** be flashed, **ISP programming ONLY**

-=- use *Code::Blocks* or the included *Makefile* to compile -=-

Author

F1RMB, Daniel Caujolle-Bert `f1rmb.daniel@gmail.com`
Lee Wiggins `lee@wigweb.com.au`

12.1.2 Function Documentation

12.1.2.1 float floatRounding (float *f*)

Float number rounding, extensively used.

Float rounding functionWe just get rid of +4 decimal values

Parameters

f	float : **Float to rounding**

Returns

float : **rounded value**

12.1.2.2 int8_t getNumericalLength (float *n*, uint8_t *len* = 3)

Get float string length, according to decimal man length.

Our float to string format function

Parameters

n	float : **Float number to analyse**
len	uint8_t : **decimal max length (3 as default)**

Returns

int8_t : **numeric value length**

12.1.2.3 int16_t serialAvailable ( )

Get the number of bytes (characters) available for reading from the serial port.

Returns

int16_t

12.1.2.4 void serialFlush ( )

Flush serial buffer (TX)

Returns

void

12.1.2.5 void serialPrint (unsigned long *n*, int *base* = DEC)

Serial printing.

Implement our serial print function to save ∼300ko

Parameters

n	unsigned long
base	int

Returns

void

12.1.2.6 void serialPrint (const char *str[]*)

Serial printing.

Parameters

str	char[]

Returns

void

12.1.2.7 void serialPrint (char *c*)

Serial printing.

Parameters

c	char

Returns

void

12.1.2.8 void serialPrint (int *n*, int16_t *base* = `DEC`)

Serial printing.

Parameters

n	int
base	int16_t

Returns

void

12.1.2.9 void serialPrint (double *n*, int *digits*)

Serial printing.

Parameters

n	double
digits	int

Returns

void

12.1.2.10 void serialPrintln ( )

Serial printing.

Returns

void

12.1.2.11 void serialPrintln (double *n*, int *digits*)

Serial printing.

Parameters

n	double
digits	int

Returns

void

12.1.2.12 void serialPrintln (char *c*)

Serial printing.

Parameters

c	char

Returns

void

12.1.2.13 uint8_t serialRead ( )

Reads incoming serial data.

Returns

uint8_t

12.1.2.14 void serialWrite (char *c*)

Serial printing.

Parameters

c	char

Returns

void

12.1.2.15 static void timer1ISR (void) [static]

ISR callback function for Timer1, used for encoder.

Returns

void

12.1.2.16 static void timer3ISR (void) [static]

ISR callback function for Timer3, used for pulse feature.

Returns

void

12.1.3 Variable Documentation

12.1.3.1 aDCEngine* pThis = NULL

12.2 aDCLoad.h File Reference

```
#include <float.h>
#include <LiquidCrystal.h>
#include <ClickEncoder.h>
#include <EEPROM.h>
```

Classes

- class aStepper

  Class that handles increase/decrease step multiplier.
- class aDCSettings

  Class that handles settings.
- struct aDCSettings::CalibrationData_t

  Calibration values.
- union aDCSettings::_eepromCalibrationValue_t

  Union to manipulate float/uint8_t [] calibration values.
- class aLCD

  LiquidDisplay extension class.
- class aDCDisplay

  Class that handles LCD displaying.
- class aInputRelay

  Class that handles input relay.
- class aDCEngine

  Main class.

Macros

- #define MAX_POWER 1

  Define this if you want 192W support (otherwise 50W)
- #define HAS_INPUT_RELAY 1

  Define this if you want input relay support.
- #define HAS_TWIN_MOSFET 1

  Define this if you're using two BUK MosFETs.

Variables

- static const uint8_t ADC_CHIPSELECT_PIN = 8

  set pin 8 as the chip select for the ADC:
- static const uint8_t ADC_INPUTVOLTAGE_CHAN = 0

  set the ADC channel that reads the input voltage.
- static const uint8_t ADC_MEASUREDCURRENT_CHAN = 1

  set the ADC channel that reads the input current by measuring the voltage on the input side of the sense resistors.
- static const uint8_t ADC_TEMPSENSE1_CHAN = 2

  set the ADC channel that reads the temperature sensor 1 under the heatsink.
- static const uint8_t ADC_TEMPSENSE2_CHAN = 3

set the ADC channel that reads the temperature sensor 2 under the heatsink.

- static const uint8_t DAC_CHIPSELECT_PIN = 9

  set pin 9 as the chip select for the DAC:
- static const uint8_t DAC_CURRENT_CHAN = 0

  set The DAC channel that sets the constant current.
- static const uint8_t DAC_FAN_CHAN = 1

  set The DAC channel that sets the fan speed.
- static const uint8_t LCD_RS_PIN = 10

  LCD RS pin.
- static const uint8_t LCD_ENABLE_PIN = 12

  LCD ENABLE pin.
- static const uint8_t LCD_D4_PIN = A0

  LCD d4 pin.
- static const uint8_t LCD_D5_PIN = A1

  LCD d5 pin.
- static const uint8_t LCD_D6_PIN = A2

  LCD d6 pin.
- static const uint8_t LCD_D7_PIN = A3

  LCD d7 pin.
- static const uint8_t LCD_COLS_NUM = 20

  LCD columns size.
- static const uint8_t LCD_ROWS_NUM = 4

  LCD rows size.
- static const uint8_t INPUT_RELAY_BUTTON_PIN = 13

  Input Relay command button.
- static const uint8_t INPUT_RELAY_RELAY_PIN = 4

  Input Relay command pin.
- static const uint8_t ENCODER_A_PIN = 3

  Encoder Channel A pin, INT 0.
- static const uint8_t ENCODER_B_PIN = 2

  Encoder Channel B pin, INT 1.
- static const uint8_t ENCODER_PB_PIN = 0

  Encoder push button pin, INT 2.
- static const uint8_t ENCODER_STEPS_PER_NOTCH = 4

  Depending on the type of your encoder, you can define use the constructors parameter `stepsPerNotch` *an set it to either* `1`, `2` *or* `4` *steps per notch, with* `1` *being the default.*
- static const uint8_t LED_BACKLIGHT_PIN = 11

  LCD backlight pin.
- static const uint8_t OFFSET_UNIT = 1

  Unit column LCD offset.
- static const uint8_t OFFSET_VALUE = 4

  Value column LCD offset.
- static const uint8_t OFFSET_TEMP = 12

  Temperature column LCD offset.
- static const uint8_t OFFSET_MARKER_LEFT = 0

  Column LCD offset for left marker '['.
- static const uint8_t OFFSET_MARKER_RIGHT = 14

  Column LCD offset for right marker ']'.
- static const uint8_t OFFSET_SETUP_MARKER_LEFT = 1

  Column LCD offset for left marker '[' in setup mode.
- static const uint8_t OFFSET_SETUP_MARKER_RIGHT = 18

Column LCD offset for right marker ']' in setup mode.

- static const uint8_t LOGGING_ICON_X_COORD = 17

  Logging icon LCD X coord.
- static const uint8_t LOGGING_ICON_Y_COORD = 3

  Logging icon LCD Y coord.
- static const uint8_t USB_ICON_X_COORD = 18

  USB icon LCD X coord.
- static const uint8_t USB_ICON_Y_COORD = 3

  USB icon LCD Y coord.
- static const uint8_t LOCK_ICON_X_COORD = 19

  LOCK icon LCD X coord.
- static const uint8_t LOCK_ICON_Y_COORD = 3

  LOCK icon LCD Y coord.
- static const uint8_t ALARM_OV_X_COORD = 18

  Over-voltage 'OV' text LCD X coord.
- static const uint8_t ALARM_OV_Y_COORD = 1

  Over-voltage 'OV' text LCD Y coord.
- static const uint8_t ALARM_OC_X_COORD = 18

  Over-current 'OC' text LCD X coord.
- static const uint8_t ALARM_OC_Y_COORD = 2

  Over-current 'OC' text LCD Y coord.
- static const uint8_t ALARM_OT_X_COORD = 18

  Over-temperature 'OT' text LCD X coord.
- static const uint8_t ALARM_OT_Y_COORD = 2

  Over-temperature 'OT' text LCD Y coord.
- static const unsigned long AUTOLOCK_TIMEOUT = 60000

  Autolock timeout value (60 seconds)
- static const unsigned long OPERATION_SET_TIMEOUT = 3000

  Automatic toggle settings->reading timeout (3 seconds)
- static const unsigned long BACKLIGHT_TIMEOUT = 600000

  Backlight dimmer timeout (10 minutes)
- static const unsigned long LOGGING_RATE = 100

  CSV data-logging rate (ms)
- static const unsigned long DISPLAY_UPDATE_RATE = 200

  Display update rate (ms)
- static const float VOLTAGE_MAXIMUM = 24.000

  Maximum handled voltage (V)
- static const float CURRENT_MAXIMUM = 7.845

  Maximum value of load current (A)
- static const float POWER_MAXIMUM = VOLTAGE_MAXIMUM ∗ CURRENT_MAXIMUM

  Maximum power dissipated (W)
- static const float PULSE_MAXIMUM = 8.192

  Maximum pulse value (ms) (unsigned 16-bit)
- static const uint16_t TEMPERATURE_MAXIMUM = 80

  Over-temperature threshold.
- static const int8_t SOFTWARE_VERSION_MAJOR = 2

  Software major version.
- static const int8_t SOFTWARE_VERSION_MINOR = 6

  Software minor version.
- static const uint16_t FEATURE_LOGGING = 1

  Bitfield logging feature.

- static const uint16_t FEATURE_LOGGING_VISIBLE = 1 << 1

  Bitfield logging icon feature.
- static const uint16_t FEATURE_USB = 1 << 2

  Bitfield USB feature.
- static const uint16_t FEATURE_USB_VISIBLE = 1 << 3

  Bitfield USB icon feature.
- static const uint16_t FEATURE_LOCKED = 1 << 4

  Bitfield locking feature.
- static const uint16_t FEATURE_LOCKED_VISIBLE = 1 << 5

  Bitfield locking icon feature.
- static const uint16_t FEATURE_AUTOLOCK = 1 << 6

  Bitfield auto-lock setting feature.
- static const uint16_t FEATURE_AUTOLOCK_VISIBLE = 1 << 7

  Bitfield auto-lock setting icon feature.
- static const uint16_t FEATURE_AUTODIM = 1 << 8

  Bitfield auto-dimmer setting feature.
- static const uint16_t FEATURE_AUTODIM_VISIBLE = 1 << 9

  Bitfield auto-dimmer setting icon feature.
- static const uint16_t FEATURE_OVP = 1 << 10

  Bitfield over-voltage protection feature.
- static const uint16_t FEATURE_OVP_VISIBLE = 1 << 11

  Bitfield over-voltage protection icon feature.
- static const uint16_t FEATURE_OCP = 1 << 12

  Bitfield over-current protection feature.
- static const uint16_t FEATURE_OCP_VISIBLE = 1 << 13

  Bitfield over-current protection icon feature.
- static const uint16_t FEATURE_OTP = 1 << 14

  Bitfield over-temperature protection feature.
- static const uint16_t FEATURE_OTP_VISIBLE = 1 << 15

  Bitfield over-temperature protection icon feature.
- static const uint8_t GLYPH_X1 = 0

  Offset of ∗1 icon in LCD HD44780 controller memory.
- static const uint8_t GLYPH_X10 = 1

  Offset of ∗10 icon in LCD HD44780 controller memory.
- static const uint8_t GLYPH_X100 = 2

  Offset of ∗100 icon in LCD HD44780 controller memory.
- static const uint8_t GLYPH_X1000 = 3

  Offset of ∗1000 icon in LCD HD44780 controller memory.
- static const uint8_t GLYPH_USB = 4

  Offset of USB icon in LCD HD44780 controller memory.
- static const uint8_t GLYPH_LOCK = 5

  Offset of LOCK icon in LCD HD44780 controller memory.
- static const uint8_t GLYPH_CHECKBOX_UNTICKED = 6

  Offset of unticked checkbox icon in LCD HD44780 controller memory.
- static const uint8_t GLYPH_CHECKBOX_TICKED = 7

  Offset of ticked checkbox icon in LCD HD44780 controller memory.
- static const int16_t EEPROM_ADDR_MAGIC = 0

  EEPROM start offset to magic numbers (0xDEAD)
- static const int16_t EEPROM_ADDR_AUTODIM = 4

  EEPROM start offset for autodimming setting.
- static const int16_t EEPROM_ADDR_AUTOLOCK = 5

EEPROM start offset for autolocking setting.

- static const int16_t EEPROM_CALIBRATION_SIZE = (sizeof(float) ∗ 2) + sizeof(uint8_t)

  EEPROM calibration size: 2 float (slope & offset), and one uint8_t for crc.

- static const int16_t EEPROM_ADDR_CALIBRATION_VOLTAGE = EEPROM_ADDR_AUTOLOCK + 1

  EEPROM start offset for voltage calibration values.

- static const int16_t EEPROM_ADDR_CALIBRATION_READ_CURRENT = EEPROM_ADDR_CALIBRATION_VOLTAGE + EEPROM_CALIBRATION_SIZE

  EEPROM start offset for current calibration values.

- static const int16_t EEPROM_ADDR_CALIBRATION_DAC_CURRENT = EEPROM_ADDR_CALIBRATION_READ_CURRENT + EEPROM_CALIBRATION_SIZE

  EEPROM start offset for DAC calibration values.

- static const int16_t EEPROM_ADDR_CALIBRATION_VOLTAGE_DROP = EEPROM_ADDR_CALIBRATION_DAC_CURRENT + EEPROM_CALIBRATION_SIZE

  EEPROM start offset for DAC calibration values.

12.2.1 Detailed Description

Copyright

Copyright (C) 2014-2015 F1RMB, Daniel Caujolle-Bert `f1rmb.daniel@gmail.com`
Copyright (C) 2014 Lee Wiggins `lee@wigweb.com.au`

License:

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Author

F1RMB, Daniel Caujolle-Bert `f1rmb.daniel@gmail.com`
Lee Wiggins `lee@wigweb.com.au`

12.2.2 Macro Definition Documentation

12.2.2.1 #define HAS_INPUT_RELAY 1

Define this if you want input relay support.

12.2.2.2 #define HAS_TWIN_MOSFET 1

Define this if you're using two BUK MosFETs.

12.2.2.3 #define MAX_POWER 1

Define this if you want 192W support (otherwise 50W)

12.2.3 Variable Documentation

12.2.3.1 const uint8_t ADC_CHIPSELECT_PIN = 8 [static]

set pin 8 as the chip select for the ADC:

Define this if you want to display Resistance settings. Undefine this to enable PULSE feature.

12.2.3.2 const uint8_t ADC_INPUTVOLTAGE_CHAN = 0 [static]

set the ADC channel that reads the input voltage.

12.2.3.3 const uint8_t ADC_MEASUREDCURRENT_CHAN = 1 [static]

set the ADC channel that reads the input current by measuring the voltage on the input side of the sense resistors.

12.2.3.4 const uint8_t ADC_TEMPSENSE1_CHAN = 2 [static]

set the ADC channel that reads the temperature sensor 1 under the heatsink.

12.2.3.5 const uint8_t ADC_TEMPSENSE2_CHAN = 3 [static]

set the ADC channel that reads the temperature sensor 2 under the heatsink.

12.2.3.6 const uint8_t ALARM_OC_X_COORD = 18 [static]

Over-current 'OC' text LCD X coord.

12.2.3.7 const uint8_t ALARM_OC_Y_COORD = 2 [static]

Over-current 'OC' text LCD Y coord.

12.2.3.8 const uint8_t ALARM_OT_X_COORD = 18 [static]

Over-temperature 'OT' text LCD X coord.

12.2.3.9 const uint8_t ALARM_OT_Y_COORD = 2 [static]

Over-temperature 'OT' text LCD Y coord.

12.2.3.10 const uint8_t ALARM_OV_X_COORD = 18 [static]

Over-voltage 'OV' text LCD X coord.

12.2.3.11 const uint8_t ALARM_OV_Y_COORD = 1 [static]

Over-voltage 'OV' text LCD Y coord.

12.2.3.12 const unsigned long AUTOLOCK_TIMEOUT = 60000 [static]

Autolock timeout value (60 seconds)

12.2.3.13 const unsigned long BACKLIGHT_TIMEOUT = 600000 [static]

Backlight dimmer timeout (10 minutes)

12.2.3.14 const float CURRENT_MAXIMUM = 7.845 [static]

Maximum value of load current (A)

12.2.3.15 const uint8_t DAC_CHIPSELECT_PIN = 9 [static]

set pin 9 as the chip select for the DAC:

12.2.3.16 const uint8_t DAC_CURRENT_CHAN = 0 [static]

set The DAC channel that sets the constant current.

12.2.3.17 const uint8_t DAC_FAN_CHAN = 1 [static]

set The DAC channel that sets the fan speed.

12.2.3.18 const unsigned long DISPLAY_UPDATE_RATE = 200 [static]

Display update rate (ms)

12.2.3.19 const int16_t EEPROM_ADDR_AUTODIM = 4 [static]

EEPROM start offset for autodimming setting.

12.2.3.20 const int16_t EEPROM_ADDR_AUTOLOCK = 5 [static]

EEPROM start offset for autolocking setting.

12.2.3.21 const int16_t EEPROM_ADDR_CALIBRATION_DAC_CURRENT = EEPROM_ADDR_CALIBRATION_READ_CURRENT + EEPROM_CALIBRATION_SIZE [static]

EEPROM start offset for DAC calibration values.

12.2.3.22 const int16_t EEPROM_ADDR_CALIBRATION_READ_CURRENT = EEPROM_ADDR_CALIBRATION_VOLTAGE + EEPROM_CALIBRATION_SIZE [static]

EEPROM start offset for current calibration values.

12.2.3.23 const int16_t EEPROM_ADDR_CALIBRATION_VOLTAGE = EEPROM_ADDR_AUTOLOCK + 1 [static]

EEPROM start offset for voltage calibration values.

12.2.3.24 const int16_t EEPROM_ADDR_CALIBRATION_VOLTAGE_DROP = EEPROM_ADDR_CALIBRATION_DAC_CURRENT + EEPROM_CALIBRATION_SIZE [static]

EEPROM start offset for DAC calibration values.

12.2.3.25 const int16_t EEPROM_ADDR_MAGIC = 0 [static]

EEPROM start offset to magic numbers (0xDEAD)

12.2.3.26 const int16_t EEPROM_CALIBRATION_SIZE = (sizeof(float) $*$ 2) + sizeof(uint8_t) [static]

EEPROM calibration size: 2 float (slope & offset), and one uint8_t for crc.

12.2.3.27 const uint8_t ENCODER_A_PIN = 3 [static]

Encoder Channel A pin, INT 0.

12.2.3.28 const uint8_t ENCODER_B_PIN = 2 [static]

Encoder Channel B pin, INT 1.

12.2.3.29 const uint8_t ENCODER_PB_PIN = 0 [static]

Encoder push button pin, INT 2.

12.2.3.30 const uint8_t ENCODER_STEPS_PER_NOTCH = 4 [static]

Depending on the type of your encoder, you can define use the constructors parameter `stepsPerNotch` an set it to either `1`, `2` or `4` steps per notch, with `1` being the default.

12.2.3.31 const uint16_t FEATURE_AUTODIM = 1 $<<$ 8 [static]

Bitfield auto-dimmer setting feature.

12.2.3.32 const uint16_t FEATURE_AUTODIM_VISIBLE = 1 $<<$ 9 [static]

Bitfield auto-dimmer setting icon feature.

12.2.3.33 const uint16_t FEATURE_AUTOLOCK = 1 $<<$ 6 [static]

Bitfield auto-lock setting feature.

12.2.3.34 const uint16_t FEATURE_AUTOLOCK_VISIBLE = 1 $<<$ 7 [static]

Bitfield auto-lock setting icon feature.

12.2.3.35 const uint16_t FEATURE_LOCKED = 1 $<<$ 4 [static]

Bitfield locking feature.

12.2.3.36 const uint16_t FEATURE_LOCKED_VISIBLE = 1 << 5 [static]

Bitfield locking icon feature.

12.2.3.37 const uint16_t FEATURE_LOGGING = 1 [static]

Bitfield logging feature.

12.2.3.38 const uint16_t FEATURE_LOGGING_VISIBLE = 1 << 1 [static]

Bitfield logging icon feature.

12.2.3.39 const uint16_t FEATURE_OCP = 1 << 12 [static]

Bitfield over-current protection feature.

12.2.3.40 const uint16_t FEATURE_OCP_VISIBLE = 1 << 13 [static]

Bitfield over-current protection icon feature.

12.2.3.41 const uint16_t FEATURE_OTP = 1 << 14 [static]

Bitfield over-temperature protection feature.

12.2.3.42 const uint16_t FEATURE_OTP_VISIBLE = 1 << 15 [static]

Bitfield over-temperature protection icon feature.

12.2.3.43 const uint16_t FEATURE_OVP = 1 << 10 [static]

Bitfield over-voltage protection feature.

12.2.3.44 const uint16_t FEATURE_OVP_VISIBLE = 1 << 11 [static]

Bitfield over-voltage protection icon feature.

12.2.3.45 const uint16_t FEATURE_USB = 1 << 2 [static]

Bitfield USB feature.

12.2.3.46 const uint16_t FEATURE_USB_VISIBLE = 1 << 3 [static]

Bitfield USB icon feature.

12.2.3.47 const uint8_t GLYPH_CHECKBOX_TICKED = 7 [static]

Offset of ticked checkbox icon in LCD HD44780 controller memory.

12.2.3.48 const uint8_t GLYPH_CHECKBOX_UNTICKED = 6 [static]

Offset of unticked checkbox icon in LCD HD44780 controller memory.

12.2.3.49 const uint8_t GLYPH_LOCK = 5 [static]

Offset of LOCK icon in LCD HD44780 controller memory.

12.2.3.50 const uint8_t GLYPH_USB = 4 [static]

Offset of USB icon in LCD HD44780 controller memory.

12.2.3.51 const uint8_t GLYPH_X1 = 0 [static]

Offset of ∗1 icon in LCD HD44780 controller memory.

12.2.3.52 const uint8_t GLYPH_X10 = 1 [static]

Offset of ∗10 icon in LCD HD44780 controller memory.

12.2.3.53 const uint8_t GLYPH_X100 = 2 [static]

Offset of ∗100 icon in LCD HD44780 controller memory.

12.2.3.54 const uint8_t GLYPH_X1000 = 3 [static]

Offset of ∗1000 icon in LCD HD44780 controller memory.

12.2.3.55 const uint8_t INPUT_RELAY_BUTTON_PIN = 13 [static]

Input Relay command button.

12.2.3.56 const uint8_t INPUT_RELAY_RELAY_PIN = 4 [static]

Input Relay command pin.

12.2.3.57 const uint8_t LCD_COLS_NUM = 20 [static]

LCD columns size.

12.2.3.58 const uint8_t LCD_D4_PIN = A0 [static]

LCD d4 pin.

Warning

pin 5 shouldn't be used, since Timer3 use it

12.2.3.59 const uint8_t LCD_D5_PIN = A1 [static]

LCD d5 pin.

Warning

pin 5 shouldn't be used, since Timer3 use it

12.2.3.60 const uint8_t LCD_D6_PIN = A2 [static]

LCD d6 pin.

Warning

pin 5 shouldn't be used, since Timer3 use it

12.2.3.61 const uint8_t LCD_D7_PIN = A3 [static]

LCD d7 pin.

Warning

pin 5 shouldn't be used, since Timer3 use it

12.2.3.62 const uint8_t LCD_ENABLE_PIN = 12 [static]

LCD ENABLE pin.

12.2.3.63 const uint8_t LCD_ROWS_NUM = 4 [static]

LCD rows size.

12.2.3.64 const uint8_t LCD_RS_PIN = 10 [static]

LCD RS pin.

12.2.3.65 const uint8_t LED_BACKLIGHT_PIN = 11 [static]

LCD backlight pin.

12.2.3.66 const uint8_t LOCK_ICON_X_COORD = 19 [static]

LOCK icon LCD X coord.

12.2.3.67 const uint8_t LOCK_ICON_Y_COORD = 3 [static]

LOCK icon LCD Y coord.

12.2.3.68 const uint8_t LOGGING_ICON_X_COORD = 17 [static]

Logging icon LCD X coord.

12.2.3.69 const uint8_t LOGGING_ICON_Y_COORD = 3 [static]

Logging icon LCD Y coord.

12.2.3.70 const unsigned long LOGGING_RATE = 100 [static]

CSV data-logging rate (ms)

12.2.3.71 const uint8_t OFFSET_MARKER_LEFT = 0 [static]

Column LCD offset for left marker '['.

12.2.3.72 const uint8_t OFFSET_MARKER_RIGHT = 14 [static]

Column LCD offset for right marker ']'.

12.2.3.73 const uint8_t OFFSET_SETUP_MARKER_LEFT = 1 [static]

Column LCD offset for left marker '[' in setup mode.

12.2.3.74 const uint8_t OFFSET_SETUP_MARKER_RIGHT = 18 [static]

Column LCD offset for right marker ']' in setup mode.

12.2.3.75 const uint8_t OFFSET_TEMP = 12 [static]

Temperature column LCD offset.

12.2.3.76 const uint8_t OFFSET_UNIT = 1 [static]

Unit column LCD offset.

12.2.3.77 const uint8_t OFFSET_VALUE = 4 [static]

Value column LCD offset.

12.2.3.78 const unsigned long OPERATION_SET_TIMEOUT = 3000 [static]

Automatic toggle settings->reading timeout (3 seconds)

12.2.3.79 const float POWER_MAXIMUM = VOLTAGE_MAXIMUM * CURRENT_MAXIMUM [static]

Maximum power dissipated (W)

12.2.3.80 const float PULSE_MAXIMUM = 8.192 [static]

Maximum pulse value (ms) (unsigned 16-bit)

12.2.3.81 const int8_t SOFTWARE_VERSION_MAJOR = 2 [static]

Software major version.

12.2.3.82 const int8_t SOFTWARE_VERSION_MINOR = 6 [static]

Software minor version.

12.2.3.83 const uint16_t TEMPERATURE_MAXIMUM = 80 [static]

Over-temperature threshold.

12.2.3.84 const uint8_t USB_ICON_X_COORD = 18 [static]

USB icon LCD X coord.

12.2.3.85 const uint8_t USB_ICON_Y_COORD = 3 [static]

USB icon LCD Y coord.

12.2.3.86 const float VOLTAGE_MAXIMUM = 24.000 [static]

Maximum handled voltage (V)

12.3 CDC.cpp File Reference

```
#include <CDC.cpp>
```

12.4 HardwareSerial.cpp File Reference

```
#include <HardwareSerial.cpp>
```

12.5 HID.cpp File Reference

```
#include <HID.cpp>
```

12.6 IPAddress.cpp File Reference

```
#include <IPAddress.cpp>
```

12.7 libraries.cpp File Reference

```
#include <Arduino.h>
#include "ClickEncoder.cpp"
#include "TimerOne.cpp"
#include "TimerThree.cpp"
```

12.8 main.cpp File Reference

```
#include <main.cpp>
```

12.9 new.cpp File Reference

```
#include <new.cpp>
```

12.10 Print.cpp File Reference

```
#include <Print.cpp>
```

12.11 sketch.cpp File Reference

```
#include <Arduino.h>
#include "aDCLoad.h"
```

Functions

- void setup ()
- void loop ()

Variables

- aDCEngine engine (LCD_RS_PIN, LCD_ENABLE_PIN, LCD_D4_PIN, LCD_D5_PIN, LCD_D6_PIN, LCD_D7_PIN, LCD_COLS_NUM, LCD_ROWS_NUM, INPUT_RELAY_BUTTON_PIN, INPUT_RELAY_RELAY_PIN, ENCODER_A_PIN, ENCODER_B_PIN, ENCODER_PB_PIN, ENCODER_STEPS_PER_NOTCH)

12.11.1 Function Documentation

12.11.1.1 void loop ( )

12.11.1.2 void setup ( )

12.11.2 Variable Documentation

12.11.2.1 aDCEngine engine(LCD_RS_PIN, LCD_ENABLE_PIN, LCD_D4_PIN, LCD_D5_PIN, LCD_D6_PIN, LCD_D7_PIN, LCD_COLS_NUM, LCD_ROWS_NUM, INPUT_RELAY_BUTTON_PIN, INPUT_RELAY_RELAY_PIN, ENCODER_A_PIN, ENCODER_B_PIN, ENCODER_PB_PIN, ENCODER_STEPS_PER_NOTCH)

!!!!! BIG FAT WARNING !!!!! Should be compiled with "-Os" flag. Bootloader couldn't be flashed, ISP programming ONLY -=- use Code::Blocks or the included Makefile to compile -=-

12.12 Stream.cpp File Reference

```
#include <Stream.cpp>
```

12.13 Tone.cpp File Reference

```
#include <Tone.cpp>
```

12.14 USBCore.cpp File Reference

```
#include <USBCore.cpp>
```

12.15 WInterrupts.c File Reference

```
#include <WInterrupts.c>
```

12.16 wiring.c File Reference

```
#include <wiring.c>
```

12.17 wiring_analog.c File Reference

```
#include <wiring_analog.c>
```

12.18 wiring_digital.c File Reference

```
#include <wiring_digital.c>
```

12.19 wiring_pulse.c File Reference

```
#include <wiring_pulse.c>
```

12.20 wiring_shift.c File Reference

```
#include <wiring_shift.c>
```

12.21 WMath.cpp File Reference

```
#include <WMath.cpp>
```

12.22 WString.cpp File Reference

```
#include <WString.cpp>
```

Index