

aDCLoad

2.4

Generated by Doxygen 1.8.6

Tue Jan 6 2015 17:30:51

Contents

1	Arduino Programmable Constant Current Power Resistance Load	1
2	ATmega32U4 fuses settings	3
3	User interface overview	5
4	Remote Commands	9
4.1	Get Identification	9
4.2	Current setting getter	9
4.3	Current setting setter	9
4.4	Calibration	10
4.5	DAC value setter (calibration purpose)	10
4.6	Current readed getter	10
4.7	Voltage readed getter	10
4.8	Logging enableity	10
4.9	Logging enableity	11
4.10	Return value	11
5	Logging data format	13
5.1	CSV logging format	13
6	Calibration Process	15
7	Hierarchical Index	19
7.1	Class Hierarchy	19
8	Class Index	21
8.1	Class List	21
9	File Index	23
9.1	File List	23
10	Class Documentation	25
10.1	aDCSettings::_eepromCalibrationValue_t Union Reference	25
10.1.1	Detailed Description	25

10.1.2	Member Data Documentation	25
10.1.2.1	c	25
10.1.2.2	v	25
10.2	aDCDisplay Class Reference	25
10.2.1	Detailed Description	26
10.2.2	Constructor & Destructor Documentation	26
10.2.2.1	aDCDisplay	26
10.2.2.2	~aDCDisplay	27
10.2.3	Member Function Documentation	27
10.2.3.1	_dimBacklight	27
10.2.3.2	_dimmingBacklight	27
10.2.3.3	_wakeupBacklight	27
10.2.3.4	isBacklightDimmed	27
10.2.3.5	pingBacklight	28
10.2.3.6	setup	28
10.2.3.7	showBanner	28
10.2.3.8	updateDisplay	28
10.2.3.9	updateField	28
10.2.4	Member Data Documentation	29
10.2.4.1	m_dimmed	29
10.2.4.2	m_dimmerTick	29
10.2.4.3	m_nextUpdate	29
10.2.4.4	m_Parent	29
10.3	aDCEngine Class Reference	29
10.3.1	Detailed Description	30
10.3.2	Constructor & Destructor Documentation	31
10.3.2.1	aDCEngine	31
10.3.2.2	~aDCEngine	32
10.3.3	Member Function Documentation	32
10.3.3.1	_adjustLoadCurrent	32
10.3.3.2	_getADC	32
10.3.3.3	_getInputVoltage	32
10.3.3.4	_getMeasuredCurrent	33
10.3.3.5	_getSettings	33
10.3.3.6	_getTemp	33
10.3.3.7	_handleButtonEvent	33
10.3.3.8	_handleLoggingAndRemote	33
10.3.3.9	_setDAC	33
10.3.3.10	_updateFanSpeed	34
10.3.3.11	_updateLoadCurrent	34

10.3.3.12	run	34
10.3.3.13	service	34
10.3.3.14	setup	34
10.3.4	Friends And Related Function Documentation	35
10.3.4.1	aDCDisplay	35
10.3.5	Member Data Documentation	35
10.3.5.1	m_Data	35
10.3.5.2	m_encoder	35
10.3.5.3	m_RXbuffer	35
10.3.5.4	m_RXoffset	35
10.3.5.5	RXBUFFER_MAXLEN	35
10.4	aDCSettings Class Reference	35
10.4.1	Detailed Description	39
10.4.2	Member Enumeration Documentation	39
10.4.2.1	CalibrationValues_t	39
10.4.2.2	DisplayMode_t	40
10.4.2.3	OperationMode_t	40
10.4.2.4	SelectionMode_t	40
10.4.2.5	SettingError_t	40
10.4.3	Constructor & Destructor Documentation	40
10.4.3.1	aDCSettings	40
10.4.3.2	~aDCSettings	40
10.4.4	Member Function Documentation	41
10.4.4.1	_crc8	41
10.4.4.2	_eepromCalibrationBackup	41
10.4.4.3	_eepromCalibrationRestore	41
10.4.4.4	_eepromCheckForMagicNumbers	41
10.4.4.5	_eepromReset	41
10.4.4.6	_eepromRestore	42
10.4.4.7	_eepromWriteMagicNumbers	42
10.4.4.8	_enableData	42
10.4.4.9	_enableDataCheck	42
10.4.4.10	_setValue	42
10.4.4.11	backupCalibration	43
10.4.4.12	enableAlarm	43
10.4.4.13	enableFeature	43
10.4.4.14	getCalibrationMode	43
10.4.4.15	getCalibrationValues	43
10.4.4.16	getCurrent	44
10.4.4.17	getCurrentDAC	44

10.4.4.18	getDisplayMode	44
10.4.4.19	getEncoderPosition	44
10.4.4.20	getFanSpeed	44
10.4.4.21	getOperationMode	45
10.4.4.22	getPower	45
10.4.4.23	getPrevNextMode	45
10.4.4.24	getSelectionMode	45
10.4.4.25	getTemperature	45
10.4.4.26	getVoltage	45
10.4.4.27	incEncoderPosition	46
10.4.4.28	isAutolocked	46
10.4.4.29	isDataEnabled	46
10.4.4.30	isFeatureEnabled	46
10.4.4.31	pingAutolock	46
10.4.4.32	pingOperationMode	47
10.4.4.33	restoreCalibration	47
10.4.4.34	setCalibrationMode	47
10.4.4.35	setCalibrationValues	47
10.4.4.36	setCurrent	47
10.4.4.37	setCurrentDAC	47
10.4.4.38	setDisplayMode	48
10.4.4.39	setEncoderPosition	48
10.4.4.40	setFanSpeed	48
10.4.4.41	setOperationMode	48
10.4.4.42	setPower	49
10.4.4.43	setSelectionMode	50
10.4.4.44	setTemperature	50
10.4.4.45	setVoltage	50
10.4.4.46	syncData	50
10.4.4.47	updateOperationMode	51
10.4.4.48	updateValuesFromMode	51
10.4.5	Member Data Documentation	51
10.4.5.1	DATA_CURRENT_READ	51
10.4.5.2	DATA_CURRENT_SETS	51
10.4.5.3	DATA_DISPLAY	51
10.4.5.4	DATA_ENCODER	51
10.4.5.5	DATA_IN_CALIBRATION	51
10.4.5.6	DATA_OPERATION	51
10.4.5.7	DATA_POWER_READ	52
10.4.5.8	DATA_POWER_SETS	52

10.4.5.9 DATA_SELECTION	52
10.4.5.10 DATA_TEMPERATURE	52
10.4.5.11 DATA_VOLTAGE	52
10.4.5.12 m_calibrationValues	52
10.4.5.13 m_currentDAC	52
10.4.5.14 m_datas	52
10.4.5.15 m_dispMode	52
10.4.5.16 m_encoderPos	52
10.4.5.17 m_fanSpeed	52
10.4.5.18 m_features	53
10.4.5.19 m_lockTick	53
10.4.5.20 m_mode	53
10.4.5.21 m_operationMode	53
10.4.5.22 m_operationTick	53
10.4.5.23 m_readCurrent	53
10.4.5.24 m_readPower	53
10.4.5.25 m_readTemperature	53
10.4.5.26 m_readVoltage	53
10.4.5.27 m_setsCurrent	53
10.4.5.28 m_setsPower	53
10.5 aLCD Class Reference	54
10.5.1 Detailed Description	54
10.5.2 Constructor & Destructor Documentation	54
10.5.2.1 aLCD	54
10.5.2.2 ~aLCD	55
10.5.3 Member Function Documentation	55
10.5.3.1 begin	55
10.5.3.2 clearValue	55
10.5.3.3 printCenter	55
10.5.3.4 printCenter	56
10.5.3.5 setCursor	56
10.5.4 Member Data Documentation	56
10.5.4.1 m_cols	56
10.5.4.2 m_curCol	56
10.5.4.3 m_curRow	56
10.5.4.4 m_rows	56
10.6 aStepper Class Reference	56
10.6.1 Detailed Description	57
10.6.2 Constructor & Destructor Documentation	57
10.6.2.1 aStepper	57

10.6.2.2	<code>~aStepper</code>	57
10.6.3	Member Function Documentation	58
10.6.3.1	<code>_pow</code>	58
10.6.3.2	<code>getMult</code>	58
10.6.3.3	<code>getValue</code>	58
10.6.3.4	<code>getValueFromMode</code>	58
10.6.3.5	<code>increment</code>	58
10.6.3.6	<code>isSynced</code>	59
10.6.3.7	<code>reset</code>	59
10.6.3.8	<code>sync</code>	59
10.6.4	Member Data Documentation	59
10.6.4.1	<code>m_inc</code>	59
10.6.4.2	<code>m_incPrev</code>	59
10.6.4.3	<code>MAX_VALUE</code>	59
10.7	<code>aDCSettings::CalibrationData_t</code> Struct Reference	59
10.7.1	Detailed Description	60
10.7.2	Member Data Documentation	60
10.7.2.1	<code>offset</code>	60
10.7.2.2	<code>slope</code>	60
11	File Documentation	61
11.1	<code>aDCLoad.cpp</code> File Reference	61
11.1.1	Detailed Description	62
11.1.2	Function Documentation	62
11.1.2.1	<code>floatRounding</code>	62
11.1.2.2	<code>getNumericalLength</code>	63
11.1.2.3	<code>serialAvailable</code>	64
11.1.2.4	<code>serialFlush</code>	64
11.1.2.5	<code>serialPrint</code>	64
11.1.2.6	<code>serialPrint</code>	64
11.1.2.7	<code>serialPrint</code>	64
11.1.2.8	<code>serialPrint</code>	65
11.1.2.9	<code>serialPrint</code>	65
11.1.2.10	<code>serialPrintln</code>	65
11.1.2.11	<code>serialPrintln</code>	65
11.1.2.12	<code>serialPrintln</code>	65
11.1.2.13	<code>serialRead</code>	66
11.1.2.14	<code>serialWrite</code>	66
11.2	<code>aDCLoad.h</code> File Reference	66
11.2.1	Detailed Description	70

11.2.2	Macro Definition Documentation	71
11.2.2.1	MAX_POWER	71
11.2.3	Typedef Documentation	71
11.2.3.1	ISRCallback	71
11.2.4	Variable Documentation	71
11.2.4.1	ADC_CHIPSELECT_PIN	71
11.2.4.2	ADC_INPUTVOLTAGE_CHAN	71
11.2.4.3	ADC_MEASUREDCURRENT_CHAN	71
11.2.4.4	ADC_TEMPSENSE1_CHAN	71
11.2.4.5	ADC_TEMPSENSE2_CHAN	71
11.2.4.6	ALARM_OC_X_COORD	71
11.2.4.7	ALARM_OC_Y_COORD	71
11.2.4.8	ALARM_OT_X_COORD	72
11.2.4.9	ALARM_OT_Y_COORD	72
11.2.4.10	ALARM_OV_X_COORD	72
11.2.4.11	ALARM_OV_Y_COORD	72
11.2.4.12	AUTOLOCK_TIMEOUT	72
11.2.4.13	BACKLIGHT_TIMEOUT	72
11.2.4.14	CURRENT_MAXIMUM	72
11.2.4.15	DAC_CHIPSELECT_PIN	72
11.2.4.16	DAC_CURRENT_CHAN	72
11.2.4.17	DAC_FAN_CHAN	72
11.2.4.18	DISPLAY_UPDATE_RATE	72
11.2.4.19	EEPROM_ADDR_AUTODIM	72
11.2.4.20	EEPROM_ADDR_AUTOLOCK	73
11.2.4.21	EEPROM_ADDR_CALIBRATION_DAC_CURRENT	73
11.2.4.22	EEPROM_ADDR_CALIBRATION_READ_CURRENT	73
11.2.4.23	EEPROM_ADDR_CALIBRATION_VOLTAGE	73
11.2.4.24	EEPROM_ADDR_CALIBRATION_VOLTAGE_DROP	73
11.2.4.25	EEPROM_ADDR_MAGIC	73
11.2.4.26	EEPROM_CALIBRATION_SIZE	73
11.2.4.27	ENCODER_A_PIN	73
11.2.4.28	ENCODER_B_PIN	73
11.2.4.29	ENCODER_PB_PIN	73
11.2.4.30	ENCODER_STEPS_PER_NOTCH	73
11.2.4.31	FEATURE_AUTODIM	74
11.2.4.32	FEATURE_AUTODIM_VISIBLE	74
11.2.4.33	FEATURE_AUTOLOCK	74
11.2.4.34	FEATURE_AUTOLOCK_VISIBLE	74
11.2.4.35	FEATURE_LOCKED	74

11.2.4.36 FEATURE_LOCKED_VISIBLE	74
11.2.4.37 FEATURE_LOGGING	74
11.2.4.38 FEATURE_LOGGING_VISIBLE	74
11.2.4.39 FEATURE_OCP	74
11.2.4.40 FEATURE_OCP_VISIBLE	74
11.2.4.41 FEATURE_OTP	74
11.2.4.42 FEATURE_OTP_VISIBLE	74
11.2.4.43 FEATURE_OVP	75
11.2.4.44 FEATURE_OVP_VISIBLE	75
11.2.4.45 FEATURE_USB	75
11.2.4.46 FEATURE_USB_VISIBLE	75
11.2.4.47 GLYPH_CHECKBOX_TICKED	75
11.2.4.48 GLYPH_CHECKBOX_UNTICKED	75
11.2.4.49 GLYPH_LOCK	75
11.2.4.50 GLYPH_USB	75
11.2.4.51 GLYPH_X1	75
11.2.4.52 GLYPH_X10	75
11.2.4.53 GLYPH_X100	75
11.2.4.54 GLYPH_X1000	75
11.2.4.55 LCD_COLS_NUM	76
11.2.4.56 LCD_D0_PIN	76
11.2.4.57 LCD_D1_PIN	76
11.2.4.58 LCD_D2_PIN	76
11.2.4.59 LCD_D3_PIN	76
11.2.4.60 LCD_D4_PIN	76
11.2.4.61 LCD_D5_PIN	76
11.2.4.62 LCD_D6_PIN	76
11.2.4.63 LCD_D7_PIN	76
11.2.4.64 LCD_ENABLE_PIN	76
11.2.4.65 LCD_ROWS_NUM	76
11.2.4.66 LCD_RS_PIN	76
11.2.4.67 LED_BACKLIGHT_PIN	77
11.2.4.68 LOCK_ICON_X_COORD	77
11.2.4.69 LOCK_ICON_Y_COORD	77
11.2.4.70 LOGGING_ICON_X_COORD	77
11.2.4.71 LOGGING_ICON_Y_COORD	77
11.2.4.72 LOGGING_RATE	77
11.2.4.73 OFFSET_MARKER_LEFT	77
11.2.4.74 OFFSET_MARKER_RIGHT	77
11.2.4.75 OFFSET_SETUP_MARKER_LEFT	77

11.2.4.76 OFFSET_SETUP_MARKER_RIGHT	77
11.2.4.77 OFFSET_TEMP	77
11.2.4.78 OFFSET_UNIT	77
11.2.4.79 OFFSET_VALUE	78
11.2.4.80 OPERATION_SET_TIMEOUT	78
11.2.4.81 POWER_MAXIMUM	78
11.2.4.82 SOFTWARE_VERSION_MAJOR	78
11.2.4.83 SOFTWARE_VERSION_MINOR	78
11.2.4.84 TEMPERATURE_MAXIMUM	78
11.2.4.85 USB_ICON_X_COORD	78
11.2.4.86 USB_ICON_Y_COORD	78
11.2.4.87 VOLTAGE_MAXIMUM	78
11.3 CDC.cpp File Reference	78
11.4 HardwareSerial.cpp File Reference	78
11.5 HID.cpp File Reference	78
11.6 IPAddress.cpp File Reference	79
11.7 libraries.cpp File Reference	79
11.8 main.cpp File Reference	79
11.9 new.cpp File Reference	79
11.10Print.cpp File Reference	79
11.11README.md File Reference	79
11.12sketch.cpp File Reference	79
11.12.1 Function Documentation	79
11.12.1.1 isr	79
11.12.1.2 loop	80
11.12.1.3 setup	80
11.12.2 Variable Documentation	80
11.12.2.1 engine	80
11.13Stream.cpp File Reference	80
11.14Tone.cpp File Reference	80
11.15USBCore.cpp File Reference	80
11.16WInterrupts.c File Reference	80
11.17wiring.c File Reference	80
11.18wiring_analog.c File Reference	80
11.19wiring_digital.c File Reference	80
11.20wiring_pulse.c File Reference	80
11.21wiring_shift.c File Reference	81
11.22WMath.cpp File Reference	81
11.23WString.cpp File Reference	81

Index	82
-----------------------	----

Chapter 1

Arduino Programmable Constant Current Power Resistance Load

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 use 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.

Chapter 2

ATmega32U4 fuses settings

Unlike the Arduino™ Leonardo 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 lfuse:w:0xE1:m
```

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

```
make fuses
```


Chapter 3

User interface overview





- The DC load control is done using a simple rotary encoder, which integrates a push button.
- 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** or **Power**.

- In both display modes, a double click changes the focus (delimited by '[' and ']' symbols) to the next value parameter, Current (**I**) or Power (**P**).
- 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** 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.

- 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 alarm	
Over-Voltage alarm	
Over-Temperature alarm (in place of "°C")	

- 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 enableity 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 triggered, 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 °C).

- The DC Load can be remotely controlled, see [Remote Commands](#)

Chapter 4

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**

4.1 Get Identification

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

See [Return value](#)

4.2 Current setting getter

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

See [Return value](#).

4.3 Current setting setter

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

See [Return value](#).

4.4 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 calibration datas into EEPROM.

See [Return value](#)

See [Calibration Process](#)

4.5 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](#)

4.6 Current readed getter

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

See [Return value](#).

4.7 Voltage readed getter

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

See [Return value](#).

4.8 Logging enability

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

See [Return value](#).

4.9 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](#).

4.10 Return value

:value:status:

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

Chapter 5

Logging data format

See also [Remote Commands](#)

Note

fields are comma separated

5.1 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 6

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 “*cute-com*” 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:

```
: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**. 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 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 amp-meter.

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 7

Hierarchical Index

7.1 Class Hierarchy

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

aDCSettings::_eepromCalibrationValue_t	25
aStepper	56
aDCSettings	35
aDCSettings::CalibrationData_t	59
LiquidCrystal	
aLCD	54
aDCDisplay	25
aDCEngine	29

Chapter 8

Class Index

8.1 Class List

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

aDCSettings::_eepromCalibrationValue_t	Union to manipulate float/uint8_t [] calibration values	25
aDCDisplay	Class that handle LCD displaying	25
aDCEngine	Main class	29
aDCSettings	Class that handle settings	35
aLCD	LiquidDisplay extension class	54
aStepper	Class that handle increase/decrease step multiplier	56
aDCSettings::CalibrationData_t	Calibration values	59

Chapter 9

File Index

9.1 File List

Here is a list of all files with brief descriptions:

aDCLoad.cpp	61
aDCLoad.h	66
CDC.cpp	78
HardwareSerial.cpp	78
HID.cpp	78
IPAddress.cpp	79
libraries.cpp	79
main.cpp	79
new.cpp	79
Print.cpp	79
sketch.cpp	79
Stream.cpp	80
Tone.cpp	80
USBCore.cpp	80
WInterrupts.c	80
wiring.c	80
wiring_analog.c	80
wiring_digital.c	80
wiring_pulse.c	80
wiring_shift.c	81
WMath.cpp	81
WString.cpp	81

- 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](#)

10.2.1 Detailed Description

Class that handle LCD displaying.

10.2.2 Constructor & Destructor Documentation

- 10.2.2.1 [aDCDisplay::aDCDisplay \(aDCEngine * parent, uint8_t rs, uint8_t enable, uint8_t d0, uint8_t d1, uint8_t d2, uint8_t d3, uint8_t d4, uint8_t d5, uint8_t d6, uint8_t d7, uint8_t cols, uint8_t rows \)](#)

Constructor.

Class to manage display output

Parameters

<i>parent</i>	aDCEngine* : Parent engine
<i>rs</i>	uint8_t : LCD RS pin
<i>enable</i>	uint8_t : LCD ENABLE pin
<i>d0</i>	uint8_t : LCD d0 pin
<i>d1</i>	uint8_t : LCD d1 pin
<i>d2</i>	uint8_t : LCD d2 pin
<i>d3</i>	uint8_t : LCD d3 pin
<i>d4</i>	uint8_t : LCD d4 pin
<i>d5</i>	uint8_t : LCD d5 pin
<i>d6</i>	uint8_t : LCD d6 pin
<i>d7</i>	uint8_t : LCD d7 pin
<i>cols</i>	uint8_t : LCD columns
<i>rows</i>	uint8_t : LCD rows

10.2.2.2 aDCDisplay::~~aDCDisplay ()

Destructor.

10.2.3 Member Function Documentation

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

Dim/undim backlight helper function.

Parameters

<i>up</i>	bool : dimmer direction
-----------	--------------------------------

Returns

void

10.2.3.2 void aDCDisplay::_dimmingBacklight () [private]

Backlight dimming.

Returns

void

10.2.3.3 void aDCDisplay::_wakeupBacklight () [private]

Backlight waker.

Returns

void

10.2.3.4 bool aDCDisplay::isBacklightDimmed ()

Check if backlight is dimmed.

Returns

bool

10.2.3.5 void aDCDisplay::pingBacklight ()

Reset backlight dimmer.

Returns

void

10.2.3.6 void aDCDisplay::setup ()

Setup function. Should be called before any other member.

Returns

void

10.2.3.7 void aDCDisplay::showBanner ()

Display small banner.

Returns

void

10.2.3.8 void aDCDisplay::updateDisplay ()

Update LCD display management function.

Draw/Redraw on screen datas

Returns

void

10.2.3.9 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

<i>opMode</i>	OperationMode_t : Operation mode
<i>vSet</i>	float : Settings value
<i>vRead</i>	float : Readed value
<i>row</i>	uint8_t : LCD row position

<i>unit</i>	uint8_t : Unit character
-------------	---------------------------------

Returns

void

10.2.4 Member Data Documentation**10.2.4.1 bool aDCDisplay::m_dimmed** [private]

Dimmed state storage.

10.2.4.2 unsigned long aDCDisplay::m_dimmerTick [private]

Dimmer timeout tick counter.

10.2.4.3 unsigned long aDCDisplay::m_nextUpdate [private]**10.2.4.4 aDCEngine* aDCDisplay::m_Parent** [private]Pointer to [aDCEngine](#) parent.

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

- [aDCLoad.h](#)
- [aDCLoad.cpp](#)

10.3 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, uint8_t=1)
- [~aDCEngine](#) ()
Destructor.
- void [setup](#) (ISRCallback)
Setup function, should be called before any other member.
- void [run](#) ()
Main loop function.
- void [service](#) ()
Caller callback function that manages encoder clicking and so on.

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 representation 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.
- const [aDCSettings](#) * [_getSettings](#) () const
Return a pointer to settings instantiated class.
- 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.

Static Private Attributes

- static const uint8_t [RXBUFFER_MAXLEN](#) = 64

Friends

- class [aDCDisplay](#)

10.3.1 Detailed Description

Main class.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 aDCEngine::aDCEngine (uint8_t *rs*, uint8_t *enable*, uint8_t *d0*, uint8_t *d1*, uint8_t *d2*, uint8_t *d3*, uint8_t *d4*, uint8_t *d5*, uint8_t *d6*, uint8_t *d7*, uint8_t *cols*, uint8_t *rows*, uint8_t *enca*, uint8_t *enclb*, uint8_t *encpb*, uint8_t *encsteps* = 1)

Class to manage settings

Parameters

<i>rs</i>	uint8_t : LCD RS pin
<i>enable</i>	uint8_t : LCD ENABLE pin
<i>d0</i>	uint8_t : LCD d0 pin
<i>d1</i>	uint8_t : LCD d1 pin
<i>d2</i>	uint8_t : LCD d2 pin
<i>d3</i>	uint8_t : LCD d3 pin
<i>d4</i>	uint8_t : LCD d4 pin
<i>d5</i>	uint8_t : LCD d5 pin
<i>d6</i>	uint8_t : LCD d6 pin
<i>d7</i>	uint8_t : LCD d7 pin
<i>cols</i>	uint8_t : LCD Columns number
<i>rows</i>	uint8_t : LCD Rows number
<i>enca</i>	uint8_t : Encoder A pin
<i>encb</i>	uint8_t : Encoder B pin
<i>encpb</i>	uint8_t : Encoder push button pin
<i>encsteps</i>	uint8_t : Encoder steps per notch

10.3.2.2 aDCEngine::~~aDCEngine ()

Destructor.

10.3.3 Member Function Documentation

10.3.3.1 void aDCEngine::_adjustLoadCurrent () [private]

Adjust current settings.

Returns

void

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

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

Parameters

<i>channel</i>	uint8_t : ADC channel
----------------	------------------------------

Returns

float : **readed value**

10.3.3.3 float aDCEngine::_getInputVoltage () [private]

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

Returns

float : **readed input voltage**

10.3.3.4 float aDCEngine::_getMeasuredCurrent () [private]

Function to measure the actual load current.

Returns

float : **readed current**

10.3.3.5 const aDCSettings* aDCEngine::_getSettings () const [private]

Return a pointer to settings instantiated class.

Returns

const aDCSettings*

10.3.3.6 int16_t aDCEngine::_getTemp () [private]

Function that read temperature from ADC channels.

Returns

int16_t : **temperature**

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

Function that handles button clicking.

Parameters

<i>button</i>	ClickEncoder::Button
---------------	----------------------

Returns

void

10.3.3.8 void aDCEngine::_handleLoggingAndRemote () [private]

Check and handle remote control and data logging.

Returns

void

10.3.3.9 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

<i>value</i>	uint16_t : value to send to DAC
<i>channel</i>	uint8_t : DAC channel

Returns

void

10.3.3.10 void aDCEngine::_updateFanSpeed () [private]

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

Returns

void

10.3.3.11 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

10.3.3.12 void aDCEngine::run ()

Main loop function.

Returns

void

10.3.3.13 void aDCEngine::service ()

Caller callback function that manages encoder clicking and so on.

Returns

void

10.3.3.14 void aDCEngine::setup (ISRCallback isr)

Setup function, should be called before any other member.

Parameters

<i>isr</i>	ISRCallback : pointer to callback function that may call service()
------------	---

Returns

void

10.3.4 Friends And Related Function Documentation

10.3.4.1 `friend class aDCDisplay` [`friend`]

10.3.5 Member Data Documentation

10.3.5.1 `aDCSettings aDCEngine::m_Data` [`private`]

Settings object.

10.3.5.2 `ClickEncoder* aDCEngine::m_encoder` [`private`]

Encoder object.

10.3.5.3 `uint8_t aDCEngine::m_RXbuffer[RXBUFFER_MAXLEN]` [`private`]

USB rx buffer.

10.3.5.4 `uint8_t aDCEngine::m_RXoffset` [`private`]

USB rx buffer offset counter.

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

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

- [aDCLoad.h](#)
- [aDCLoad.cpp](#)

10.4 aDCSettings Class Reference

Class that handle 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_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` ()
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 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 [setCalibrationMode](#) (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_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

- float [m_readVoltage](#)
voltage storage
- float [m_setsCurrent](#)
- float [m_readCurrent](#)
current storage
- 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 displays data storage
- [CalibrationData_t m_calibrationValues](#) [CALIBRATION_MAX]
Calibration datas, restored from EEPROM.

10.4.1 Detailed Description

Class that handle settings.

10.4.2 Member Enumeration Documentation

10.4.2.1 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.

10.4.2.2 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)

10.4.2.3 enum aDCSettings::OperationMode_t

Operation mode enumeration.

Enumerator

OPERATION_MODE_READ Reading values.
OPERATION_MODE_SET Settings values.
OPERATION_MODE_UNKNOWN Unset (internal)

10.4.2.4 enum aDCSettings::SelectionMode_t

Selection settings mode enumeration.

Enumerator

SELECTION_MODE_CURRENT Current selected.
SELECTION_MODE_POWER Power selected.
SELECTION_MODE_UNKNOWN Nothing selected (internal)

10.4.2.5 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.

10.4.3 Constructor & Destructor Documentation

10.4.3.1 aDCSettings::aDCSettings ()

Constructor.

10.4.3.2 aDCSettings::~~aDCSettings ()

Destructor.

10.4.4 Member Function Documentation

10.4.4.1 `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

<i>addr</i>	const uint8_t* : Data source
<i>len</i>	uint8_t : Data source length

Returns

uint8_t : **CRC**

10.4.4.2 `void aDCSettings::_eepromCalibrationBackup (int16_t addr, CalibrationData_t cal)` [private]

Save data to EEPROM, used to restore calibration data.

Parameters

<i>addr</i>	int16_t : start address location
<i>cal</i>	CalibrationData_t & : destination

Returns

void

10.4.4.3 `void aDCSettings::_eepromCalibrationRestore (int16_t addr, CalibrationData_t & cal)` [private]

Read data from EEPROM, used to restore calibration data.

Parameters

<i>addr</i>	int16_t : start address location
<i>cal</i>	CalibrationData_t & : destination

Returns

void

10.4.4.4 `bool aDCSettings::_eepromCheckForMagicNumbers ()` [private]

Check for EEPROM magic numbers.

Returns

bool

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

10.4.4.5 `void aDCSettings::_eepromReset ()` [private]

Reset all stored parameters into EEPROM.

Returns

void

10.4.4.6 void aDCSettings::_eepromRestore () [private]

Restore value from EEPROM.

Returns

void

10.4.4.7 void aDCSettings::_eepromWriteMagicNumbers () [private]

Write magic numbers into EEPROM.

Returns

void

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

Enable a bit in the m_datas bit-field storage.

Parameters

<i>bit</i>	uint16_t : Bit to set
<i>enable</i>	bool : Bit enableity

Returns

void

10.4.4.9 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

<i>bit</i>	uint16_t : Bit to set
<i>enable</i>	bool : Bit enableity

Returns

void

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

Value setter.

Parameters

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

Returns

SettingError_t : **validity result**

10.4.4.11 void aDCSettings::backupCalibration ()

Backup calibration data into EEPROM.

Returns

void

10.4.4.12 void aDCSettings::enableAlarm (uint16_t aBit)

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

Parameters

<i>aBit</i>	uint16_t : alarm bit to set
-------------	------------------------------------

Returns

void

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

Helper function to manage bit-field features.

Parameters

<i>feature</i>	uint16_t : FEATURE to enable/disable
<i>enable</i>	bool : FEATURE enableity (default = enable)

Returns

void

10.4.4.14 bool aDCSettings::getCalibrationMode ()

Calibration mode enableity getter.

Returns

bool : **enableity**

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

Calibration data getter, according to *calsection* argument.

Parameters

<i>calsection</i>	CalibrationValues_t : calibration parameter
<i>data</i>	CalibrationData_t & : calibration data

Returns

void

10.4.4.16 float aDCSettings::getCurrent (OperationMode_t mode)

Current getter.

Parameters

<i>mode</i>	OperationMode_t : READ/SET mode storage access
-------------	---

Returns

float : **current**

10.4.4.17 uint16_t aDCSettings::getCurrentDAC ()

Current DAC value getter.

Returns

uint16_t : **current DAC value**

10.4.4.18 aDCSettings::DisplayMode_t aDCSettings::getDisplayMode ()

Display mode getter.

Returns

DisplayMode_t : **Display mode**

10.4.4.19 int32_t aDCSettings::getEncoderPosition ()

Encoder position getter.

Returns

int32_t : **encoder position**

10.4.4.20 uint16_t aDCSettings::getFanSpeed ()

Fan speed getter.

Returns

uint16_t : **DAC speed value**

10.4.4.21 aDCSettings::OperationMode_t aDCSettings::getOperationMode ()

Operation mode getter.

Returns

OperationMode_t : **operation mode**

10.4.4.22 float aDCSettings::getPower (OperationMode_t mode)

Power getter.

Parameters

<i>mode</i>	OperationMode_t : READ/SET mode storage access
-------------	---

Returns

float : **power**

10.4.4.23 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

<i>origin</i>	SelectionMode_t : origin starter selection mode
<i>next</i>	bool : Next or Previous mode

Returns

SelectionMode_t : **next selection mode**

10.4.4.24 aDCSettings::SelectionMode_t aDCSettings::getSelectionMode ()

Selection mode getter.

Returns

SelectionMode_t : **current selection mode**

10.4.4.25 uint16_t aDCSettings::getTemperature ()

Temperature readed getter.

Returns

uint16_t : **temperature**

10.4.4.26 float aDCSettings::getVoltage ()

Voltage getter.

Returns

float : **voltage**

10.4.4.27 void aDCSettings::incEncoderPosition (int32_t *v* = 1)

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

Parameters

<i>v</i>	int32_t : increment value, 1 by default
----------	--

Returns

void

10.4.4.28 bool aDCSettings::isAutolocked ()

Check if autolock is enabled and performs.

Returns

bool

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

Get bit enability in m_datas bit-field storage.

Parameters

<i>bit</i>	uint16_t : Bit to check
------------	--------------------------------

Returns

bool

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

Check if FEATURE is enabled.

Parameters

<i>feature</i>	uint16_t : FEATURE to check against
----------------	--

Returns

bool

10.4.4.31 void aDCSettings::pingAutolock ()

Reset autolock timeout.

Returns

void

10.4.4.32 void aDCSettings::pingOperationMode ()

Reset timeout while in OPERATION_SET mode.

Returns

void

10.4.4.33 void aDCSettings::restoreCalibration ()

Restore calibration data from EEPROM.

Returns

void

10.4.4.34 void aDCSettings::setCalibrationMode (bool *enable* = true)

Calibration mode enable setter.

Parameters

<i>enable</i>	bool : enable
---------------	----------------------

Returns

void

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

Calibration data setter, according to *calsection* argument.

Parameters

<i>calsection</i>	CalibrationValues_t : calibration parameter
<i>data</i>	CalibrationData_t & : calibration data

Returns

void

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

Current setter.

Parameters

<i>v</i>	float : current
<i>mode</i>	OperationMode_t : READ/SET mode storage access

Returns

[aDCSettings::SettingError_t](#)

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

Current DAC value setter.

Parameters

<i>dac</i>	uint16_t : current DAC value
------------	-------------------------------------

Returns

void

10.4.4.38 void aDCSettings::setDisplayMode (DisplayMode_t d)

Display mode setter.

Parameters

<i>d</i>	DisplayMode_t : display mode
----------	-------------------------------------

Returns

void

10.4.4.39 void aDCSettings::setEncoderPosition (int32_t v)

Encoder position setter.

Parameters

<i>v</i>	int32_t : encoder position
----------	-----------------------------------

Returns

void

10.4.4.40 void aDCSettings::setFanSpeed (uint16_t v)

Fan speed setter.

Parameters

<i>v</i>	uint16_t : DAC speed value
----------	-----------------------------------

Returns

void

10.4.4.41 void aDCSettings::setOperationMode (OperationMode_t m)

Operation mode setter.

Parameters

<i>m</i>	OperationMode_t : new operation mode
----------	---

Returns

void

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

Power setter.

Parameters

<i>v</i>	float : power
<i>mode</i>	OperationMode_t : READ/SET mode storage access

Returns

[aDCSettings::SettingError_t](#)

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

Selection mode setter.

Parameters

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

Returns

void

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

Temperature readed setter.

Parameters

<i>v</i>	uint16_t : temperature
----------	-------------------------------

Returns

void

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

Voltage setter.

Parameters

<i>v</i>	float : voltage
----------	------------------------

Returns

[aDCSettings::SettingError_t](#)

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

Clear a bit in m_datas bit-field storage.

Parameters

<i>bit</i>	uint16_t
------------	----------

Returns

void

10.4.4.47 void aDCSettings::updateOperationMode ()

Automatic timeouted toggle between OPERATION_SET and OPERATION_READ.

Returns

void

10.4.4.48 void aDCSettings::updateValuesFromMode (float v, SelectionMode_t mode)

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

Parameters

<i>v</i>	float : updated value
<i>mode</i>	SelectionMode_t : selection mode (CURRENT, RESISTANCE, POWER)

Returns

void

10.4.5 Member Data Documentation**10.4.5.1 const uint16_t aDCSettings::DATA_CURRENT_READ = 1 << 2 [static]**

bit-field storage: Current readed

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

bit-field storage: Current sets

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

bit-field storage: Display mode sets

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

bit-field storage: Encoder position sets

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

bit-field storage: In calibration mode

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

bit-field storage: Operation mode sets

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

bit-field storage: Power readed

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

bit-field storage: Power sets

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

bit-field storage: Selection mode sets

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

bit-field storage: Temperature readed

10.4.5.11 `const uint16_t aDCSettings::DATA_VOLTAGE = 1` `[static]`

bit-field storage: Voltage readed

10.4.5.12 `CalibrationData_t aDCSettings::m_calibrationValues[CALIBRATION_MAX]` `[private]`

Calibration datas, restored from EEPROM.

10.4.5.13 `uint16_t aDCSettings::m_currentDAC` `[private]`

current DAC value

10.4.5.14 `uint16_t aDCSettings::m_datas` `[private]`

boolean displays data storage

10.4.5.15 `DisplayMode_t aDCSettings::m_dispMode` `[private]`

display mode

See Also

`DisplayMode`

10.4.5.16 `int32_t aDCSettings::m_encoderPos` `[private]`

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

10.4.5.17 `uint16_t aDCSettings::m_fanSpeed` `[private]`

fan speed storage

10.4.5.18 `uint16_t aDCSettings::m_features` [private]

boolean features storage

10.4.5.19 `unsigned long aDCSettings::m_lockTick` [private]

tick count storage, for autolock feature

10.4.5.20 `SelectionMode_t aDCSettings::m_mode` [private]

selection mode

See Also

`SelectionMode`

10.4.5.21 `OperationMode_t aDCSettings::m_operationMode` [private]

operation mode

See Also

`OperationMode`

10.4.5.22 `unsigned long aDCSettings::m_operationTick` [private]

tick count storage, for SET/READ operation

10.4.5.23 `float aDCSettings::m_readCurrent` [private]

current storage

10.4.5.24 `float aDCSettings::m_readPower` [private]

power storage

10.4.5.25 `uint16_t aDCSettings::m_readTemperature` [private]

temperature storage

10.4.5.26 `float aDCSettings::m_readVoltage` [private]

voltage storage

10.4.5.27 `float aDCSettings::m_setsCurrent` [private]

10.4.5.28 `float aDCSettings::m_setsPower` [private]

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

- [aDCLoad.h](#)
- [aDCLoad.cpp](#)

10.5 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, 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.

10.5.1 Detailed Description

LiquidDisplay extension class.

10.5.2 Constructor & Destructor Documentation

10.5.2.1 [aLCD::aLCD](#) (uint8_t *rs*, uint8_t *enable*, uint8_t *d0*, uint8_t *d1*, uint8_t *d2*, uint8_t *d3*, uint8_t *d4*, uint8_t *d5*, uint8_t *d6*, uint8_t *d7*, uint8_t *cols*, uint8_t *rows*)

Constructor.

LiquidDisplay extension

Parameters

<i>rs</i>	uint8_t : LCD RS pin
<i>enable</i>	uint8_t : LCD ENABLE pin
<i>d0</i>	uint8_t : LCD d0 pin
<i>d1</i>	uint8_t : LCD d1 pin
<i>d2</i>	uint8_t : LCD d2 pin
<i>d3</i>	uint8_t : LCD d3 pin
<i>d4</i>	uint8_t : LCD d4 pin
<i>d5</i>	uint8_t : LCD d5 pin
<i>d6</i>	uint8_t : LCD d6 pin
<i>d7</i>	uint8_t : LCD d7 pin
<i>cols</i>	uint8_t : LCD columns number
<i>rows</i>	uint8_t : LCD rows number

10.5.2.2 aLCD::~~aLCD ()

Destructor.

10.5.3 Member Function Documentation

10.5.3.1 void aLCD::begin (uint8_t cols, uint8_t lines)

Initialize LCD screen size.

Parameters

<i>cols</i>	uint8_t : Columns
<i>lines</i>	uint8_t : Rows

Returns

void

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

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

Parameters

<i>row</i>	uint8_t : field row
<i>destMinus</i>	int : minus end field position (default = 0)

Returns

void

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

Print centered string to current row.

Parameters

<i>str</i>	const char* : String to display
------------	--

Returns

void

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

Print centered string to current row.

Parameters

<i>ifsh</i>	const __FlashStringHelper* : string to display
-------------	---

Returns

void

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

Set cursor position.

Parameters

<i>col</i>	uint8_t : Column
<i>row</i>	uint8_t : Row

Returns

void

10.5.4 Member Data Documentation

10.5.4.1 uint8_t aLCD::m_cols [private]

10.5.4.2 uint8_t aLCD::m_curCol [private]

10.5.4.3 uint8_t aLCD::m_curRow [private]

Current cursor position.

10.5.4.4 uint8_t aLCD::m_rows [private]

LCD sizes.

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

- [aDCLoad.h](#)
- [aDCLoad.cpp](#)

10.6 aStepper Class Reference

Class that handle 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

10.6.1 Detailed Description

Class that handle increase/decrease step multiplier.

Classes declarations

10.6.2 Constructor & Destructor Documentation

10.6.2.1 [aStepper::aStepper](#) ()

Constructor.

10.6.2.2 [aStepper::~~aStepper](#) ()

Destructor.

10.6.3 Member Function Documentation

10.6.3.1 `int16_t aStepper::_pow (int base, int exp)` `[inline], [private]`

Small implementation of pow() math function.

Parameters

<i>base</i>	int : base radix
<i>exp</i>	int : exponent value

Returns

`int16_t` : **result**

10.6.3.2 `int16_t aStepper::getMult ()`

Value getter, according to multiple.

Returns

`int16_t`

10.6.3.3 `uint8_t aStepper::getValue ()`

Value getter.

Returns

`uint8_t`

10.6.3.4 `int16_t aStepper::getValueFromMode (uint8_t mode)`

Get value according to selection mode.

Parameters

<i>mode</i>	<code>uint16_t</code> : Selection mode (will be typecasted to aDCSettings::SelectionMode_t)
-------------	--

Returns

`int16_t`

10.6.3.5 `void aStepper::increment ()`

Increment value, check for boundaries.

Returns

`void`

10.6.3.6 bool aStepper::isSynced ()

Synchronize value.

Returns

bool

10.6.3.7 void aStepper::reset ()

Reset value.

Returns

void

10.6.3.8 void aStepper::sync ()

Check if value is synchronized.

Returns

void

10.6.4 Member Data Documentation

10.6.4.1 uint8_t aStepper::m_inc [private]

10.6.4.2 uint8_t aStepper::m_incPrev [private]

Stepper counters.

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

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

- [aDCLoad.h](#)
- [aDCLoad.cpp](#)

10.7 aDCSettings::CalibrationData_t Struct Reference

Calibration values.

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

Public Attributes

- float [slope](#)
Slope value.
- float [offset](#)
Offset value.

10.7.1 Detailed Description

Calibration values.

Contains Slope and Offset float values

10.7.2 Member Data Documentation

10.7.2.1 float aDCSettings::CalibrationData_t::offset

Offset value.

10.7.2.2 float aDCSettings::CalibrationData_t::slope

Slope value.

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

- [aDCLoad.h](#)

Chapter 11

File Documentation

11.1 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 "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.

11.1.1 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

11.1.2 Function Documentation

11.1.2.1 float floatRounding (float f)

Float number rounding, extensively used.

Float rounding functionWe just get rid of +4 decimal values

Parameters

<i>f</i>	float : Float to rounding
----------	----------------------------------

Returns

float : **rounded value**

11.1.2.2 `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

<i>n</i>	float : Float number to analyse
<i>len</i>	uint8_t : decimal max length (3 as default)

Returns

int8_t : **numeric value length**

11.1.2.3 int16_t serialAvailable ()

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

Returns

int16_t

11.1.2.4 void serialFlush ()

Flush serial buffer (TX)

Returns

void

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

Serial printing.

Implement our serial print function to save ~300ko

Parameters

<i>n</i>	unsigned long
<i>base</i>	int

Returns

void

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

Serial printing.

Parameters

<i>str</i>	char[]
------------	--------

Returns

void

11.1.2.7 void serialPrint (char *c*)

Serial printing.

Parameters

<i>c</i>	char
----------	------

Returns

void

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

Serial printing.

Parameters

<i>n</i>	int
<i>base</i>	int16_t

Returns

void

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

Serial printing.

Parameters

<i>n</i>	double
<i>digits</i>	int

Returns

void

11.1.2.10 void serialPrintln ()

Serial printing.

Returns

void

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

Serial printing.

Parameters

<i>n</i>	double
<i>digits</i>	int

Returns

void

11.1.2.12 void serialPrintln (char *c*)

Serial printing.

Parameters

<code>c</code>	<code>char</code>
----------------	-------------------

Returns

`void`

11.1.2.13 `uint8_t serialRead ()`

Reads incoming serial data.

Returns

`uint8_t`

11.1.2.14 `void serialWrite (char c)`

Serial printing.

Parameters

<code>c</code>	<code>char</code>
----------------	-------------------

Returns

`void`

11.2 aDCLoad.h File Reference

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

Classes

- class [aStepper](#)
Class that handle increase/decrease step multiplier.
- class [aDCSettings](#)
Class that handle 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 handle LCD displaying.
- class [aDCEngine](#)
Main class.

Macros

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

Typedefs

- `typedef void(* ISRCallback)()`
Function prototype for ISR callback.

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_D0_PIN = A0`
LCD d0 pin.
- `static const uint8_t LCD_D1_PIN = A1`
LCD d1 pin.
- `static const uint8_t LCD_D2_PIN = A2`
LCD d3 pin.
- `static const uint8_t LCD_D3_PIN = A3`
LCD d2 pin.
- `static const uint8_t LCD_D4_PIN = 4`
LCD d4 pin.
- `static const uint8_t LCD_D5_PIN = 13`
LCD d5 pin.
- `static const uint8_t LCD_D6_PIN = 6`
LCD d6 pin.
- `static const uint8_t LCD_D7_PIN = 5`
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 `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 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` = 4
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.*

11.2.1 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

11.2.2 Macro Definition Documentation

11.2.2.1 #define MAX_POWER 1

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

11.2.3 Typedef Documentation

11.2.3.1 typedef void(* ISRCallback)()

Function prototype for ISR callback.

11.2.4 Variable Documentation

11.2.4.1 const uint8_t ADC_CHIPSELECT_PIN = 8 [static]

set pin 8 as the chip select for the ADC:

11.2.4.2 const uint8_t ADC_INPUTVOLTAGE_CHAN = 0 [static]

set the ADC channel that reads the input voltage.

11.2.4.3 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.

11.2.4.4 const uint8_t ADC_TEMPSENSE1_CHAN = 2 [static]

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

11.2.4.5 const uint8_t ADC_TEMPSENSE2_CHAN = 3 [static]

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

11.2.4.6 const uint8_t ALARM_OC_X_COORD = 18 [static]

Over-current 'OC' text LCD X coord.

11.2.4.7 const uint8_t ALARM_OC_Y_COORD = 2 [static]

Over-current 'OC' text LCD Y coord.

11.2.4.8 `const uint8_t ALARM_OT_X_COORD = 18` [static]

Over-temperature 'OT' text LCD X coord.

11.2.4.9 `const uint8_t ALARM_OT_Y_COORD = 2` [static]

Over-temperature 'OT' text LCD Y coord.

11.2.4.10 `const uint8_t ALARM_OV_X_COORD = 18` [static]

Over-voltage 'OV' text LCD X coord.

11.2.4.11 `const uint8_t ALARM_OV_Y_COORD = 1` [static]

Over-voltage 'OV' text LCD Y coord.

11.2.4.12 `const unsigned long AUTOLOCK_TIMEOUT = 60000` [static]

Autolock timeout value (60 seconds)

11.2.4.13 `const unsigned long BACKLIGHT_TIMEOUT = 600000` [static]

Backlight dimmer timeout (10 minutes)

11.2.4.14 `const float CURRENT_MAXIMUM = 7.845` [static]

Maximum value of load current (A)

11.2.4.15 `const uint8_t DAC_CHIPSELECT_PIN = 9` [static]

set pin 9 as the chip select for the DAC:

11.2.4.16 `const uint8_t DAC_CURRENT_CHAN = 0` [static]

set The DAC channel that sets the constant current.

11.2.4.17 `const uint8_t DAC_FAN_CHAN = 1` [static]

set The DAC channel that sets the fan speed.

11.2.4.18 `const unsigned long DISPLAY_UPDATE_RATE = 200` [static]

Display update rate (ms)

11.2.4.19 `const int16_t EEPROM_ADDR_AUTODIM = 4` [static]

EEPROM start offset for autodimming setting.

11.2.4.20 `const int16_t EEPROM_ADDR_AUTOLOCK = 5 [static]`

EEPROM start offset for autolocking setting.

11.2.4.21 `const int16_t EEPROM_ADDR_CALIBRATION_DAC_CURRENT = EEPROM_ADDR_CALIBRATION_READ_CURRENT + EEPROM_CALIBRATION_SIZE [static]`

EEPROM start offset for DAC calibration values.

11.2.4.22 `const int16_t EEPROM_ADDR_CALIBRATION_READ_CURRENT = EEPROM_ADDR_CALIBRATION_VOLTAGE + EEPROM_CALIBRATION_SIZE [static]`

EEPROM start offset for current calibration values.

11.2.4.23 `const int16_t EEPROM_ADDR_CALIBRATION_VOLTAGE = EEPROM_ADDR_AUTOLOCK + 1 [static]`

EEPROM start offset for voltage calibration values.

11.2.4.24 `const int16_t EEPROM_ADDR_CALIBRATION_VOLTAGE_DROP = EEPROM_ADDR_CALIBRATION_DAC_CURRENT + EEPROM_CALIBRATION_SIZE [static]`

EEPROM start offset for DAC calibration values.

11.2.4.25 `const int16_t EEPROM_ADDR_MAGIC = 0 [static]`

EEPROM start offset to magic numbers (0xDEAD)

11.2.4.26 `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.

11.2.4.27 `const uint8_t ENCODER_A_PIN = 3 [static]`

Encoder Channel A pin, INT 0.

11.2.4.28 `const uint8_t ENCODER_B_PIN = 2 [static]`

Encoder Channel B pin, INT 1.

11.2.4.29 `const uint8_t ENCODER_PB_PIN = 0 [static]`

Encoder push button pin, INT 2.

11.2.4.30 `const uint8_t ENCODER_STEPS_PER_NOTCH = 4 [static]`

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

11.2.4.31 `const uint16_t FEATURE_AUTODIM = 1 << 8` [static]

Bitfield auto-dimmer setting feature.

11.2.4.32 `const uint16_t FEATURE_AUTODIM_VISIBLE = 1 << 9` [static]

Bitfield auto-dimmer setting icon feature.

11.2.4.33 `const uint16_t FEATURE_AUTOLOCK = 1 << 6` [static]

Bitfield auto-lock setting feature.

11.2.4.34 `const uint16_t FEATURE_AUTOLOCK_VISIBLE = 1 << 7` [static]

Bitfield auto-lock setting icon feature.

11.2.4.35 `const uint16_t FEATURE_LOCKED = 1 << 4` [static]

Bitfield locking feature.

11.2.4.36 `const uint16_t FEATURE_LOCKED_VISIBLE = 1 << 5` [static]

Bitfield locking icon feature.

11.2.4.37 `const uint16_t FEATURE_LOGGING = 1` [static]

Bitfield logging feature.

11.2.4.38 `const uint16_t FEATURE_LOGGING_VISIBLE = 1 << 1` [static]

Bitfield logging icon feature.

11.2.4.39 `const uint16_t FEATURE_OCP = 1 << 12` [static]

Bitfield over-current protection feature.

11.2.4.40 `const uint16_t FEATURE_OCP_VISIBLE = 1 << 13` [static]

Bitfield over-current protection icon feature.

11.2.4.41 `const uint16_t FEATURE_OTP = 1 << 14` [static]

Bitfield over-temperature protection feature.

11.2.4.42 `const uint16_t FEATURE_OTP_VISIBLE = 1 << 15` [static]

Bitfield over-temperature protection icon feature.

11.2.4.43 `const uint16_t FEATURE_OVP = 1 << 10` `[static]`

Bitfield over-voltage protection feature.

11.2.4.44 `const uint16_t FEATURE_OVP_VISIBLE = 1 << 11` `[static]`

Bitfield over-voltage protection icon feature.

11.2.4.45 `const uint16_t FEATURE_USB = 1 << 2` `[static]`

Bitfield USB feature.

11.2.4.46 `const uint16_t FEATURE_USB_VISIBLE = 1 << 3` `[static]`

Bitfield USB icon feature.

11.2.4.47 `const uint8_t GLYPH_CHECKBOX_TICKED = 7` `[static]`

Offset of ticked checkbox icon in LCD HD44780 controller memory.

11.2.4.48 `const uint8_t GLYPH_CHECKBOX_UNTICKED = 6` `[static]`

Offset of unticked checkbox icon in LCD HD44780 controller memory.

11.2.4.49 `const uint8_t GLYPH_LOCK = 5` `[static]`

Offset of LOCK icon in LCD HD44780 controller memory.

11.2.4.50 `const uint8_t GLYPH_USB = 4` `[static]`

Offset of USB icon in LCD HD44780 controller memory.

11.2.4.51 `const uint8_t GLYPH_X1 = 0` `[static]`

Offset of *1 icon in LCD HD44780 controller memory.

11.2.4.52 `const uint8_t GLYPH_X10 = 1` `[static]`

Offset of *10 icon in LCD HD44780 controller memory.

11.2.4.53 `const uint8_t GLYPH_X100 = 2` `[static]`

Offset of *100 icon in LCD HD44780 controller memory.

11.2.4.54 `const uint8_t GLYPH_X1000 = 3` `[static]`

Offset of *1000 icon in LCD HD44780 controller memory.

11.2.4.55 `const uint8_t LCD_COLS_NUM = 20` `[static]`

LCD columns size.

11.2.4.56 `const uint8_t LCD_D0_PIN = A0` `[static]`

LCD d0 pin.

11.2.4.57 `const uint8_t LCD_D1_PIN = A1` `[static]`

LCD d1 pin.

11.2.4.58 `const uint8_t LCD_D2_PIN = A2` `[static]`

LCD d3 pin.

11.2.4.59 `const uint8_t LCD_D3_PIN = A3` `[static]`

LCD d2 pin.

11.2.4.60 `const uint8_t LCD_D4_PIN = 4` `[static]`

LCD d4 pin.

11.2.4.61 `const uint8_t LCD_D5_PIN = 13` `[static]`

LCD d5 pin.

11.2.4.62 `const uint8_t LCD_D6_PIN = 6` `[static]`

LCD d6 pin.

11.2.4.63 `const uint8_t LCD_D7_PIN = 5` `[static]`

LCD d7 pin.

11.2.4.64 `const uint8_t LCD_ENABLE_PIN = 12` `[static]`

LCD ENABLE pin.

11.2.4.65 `const uint8_t LCD_ROWS_NUM = 4` `[static]`

LCD rows size.

11.2.4.66 `const uint8_t LCD_RS_PIN = 10` `[static]`

LCD RS pin.

11.2.4.67 `const uint8_t LED_BACKLIGHT_PIN = 11` `[static]`

LCD backlight pin.

11.2.4.68 `const uint8_t LOCK_ICON_X_COORD = 19` `[static]`

LOCK icon LCD X coord.

11.2.4.69 `const uint8_t LOCK_ICON_Y_COORD = 3` `[static]`

LOCK icon LCD Y coord.

11.2.4.70 `const uint8_t LOGGING_ICON_X_COORD = 17` `[static]`

Logging icon LCD X coord.

11.2.4.71 `const uint8_t LOGGING_ICON_Y_COORD = 3` `[static]`

Logging icon LCD Y coord.

11.2.4.72 `const unsigned long LOGGING_RATE = 100` `[static]`

CSV data-logging rate (ms)

11.2.4.73 `const uint8_t OFFSET_MARKER_LEFT = 0` `[static]`

Column LCD offset for left marker '['.

11.2.4.74 `const uint8_t OFFSET_MARKER_RIGHT = 14` `[static]`

Column LCD offset for right marker ']'.

11.2.4.75 `const uint8_t OFFSET_SETUP_MARKER_LEFT = 1` `[static]`

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

11.2.4.76 `const uint8_t OFFSET_SETUP_MARKER_RIGHT = 18` `[static]`

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

11.2.4.77 `const uint8_t OFFSET_TEMP = 12` `[static]`

Temperature column LCD offset.

11.2.4.78 `const uint8_t OFFSET_UNIT = 1` `[static]`

Unit column LCD offset.

11.2.4.79 `const uint8_t OFFSET_VALUE = 4` `[static]`

Value column LCD offset.

11.2.4.80 `const unsigned long OPERATION_SET_TIMEOUT = 3000` `[static]`

Automatic toggle settings->reading timeout (3 seconds)

11.2.4.81 `const float POWER_MAXIMUM = VOLTAGE_MAXIMUM * CURRENT_MAXIMUM` `[static]`

Maximum power dissipated (W)

11.2.4.82 `const int8_t SOFTWARE_VERSION_MAJOR = 2` `[static]`

Software major version.

11.2.4.83 `const int8_t SOFTWARE_VERSION_MINOR = 4` `[static]`

Software minor version.

11.2.4.84 `const uint16_t TEMPERATURE_MAXIMUM = 80` `[static]`

Over-temperature threshold.

11.2.4.85 `const uint8_t USB_ICON_X_COORD = 18` `[static]`

USB icon LCD X coord.

11.2.4.86 `const uint8_t USB_ICON_Y_COORD = 3` `[static]`

USB icon LCD Y coord.

11.2.4.87 `const float VOLTAGE_MAXIMUM = 24.000` `[static]`

Maximum handled voltage (V)

11.3 CDC.cpp File Reference

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

11.4 HardwareSerial.cpp File Reference

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

11.5 HID.cpp File Reference

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

11.6 IPAddress.cpp File Reference

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

11.7 libraries.cpp File Reference

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

11.8 main.cpp File Reference

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

11.9 new.cpp File Reference

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

11.10 Print.cpp File Reference

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

11.11 README.md File Reference

11.12 sketch.cpp File Reference

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

Functions

- void [isr](#) ()
- void [setup](#) ()
- void [loop](#) ()

Variables

- [aDCEngine engine](#) ([LCD_RS_PIN](#), [LCD_ENABLE_PIN](#), [LCD_D0_PIN](#), [LCD_D1_PIN](#), [LCD_D2_PIN](#), [LCD_D3_PIN](#), [LCD_D4_PIN](#), [LCD_D5_PIN](#), [LCD_D6_PIN](#), [LCD_D7_PIN](#), [LCD_COLS_NUM](#), [LCD_ROWS_NUM](#), [ENCODER_A_PIN](#), [ENCODER_B_PIN](#), [ENCODER_PB_PIN](#), [ENCODER_STEPS_PER_NOTCH](#))

11.12.1 Function Documentation

11.12.1.1 void [isr](#) ()

11.12.1.2 `void loop ()`

11.12.1.3 `void setup ()`

11.12.2 Variable Documentation

11.12.2.1 `aDCEngine engine(LCD_RS_PIN, LCD_ENABLE_PIN, LCD_D0_PIN, LCD_D1_PIN, LCD_D2_PIN, LCD_D3_PIN, LCD_D4_PIN, LCD_D5_PIN, LCD_D6_PIN, LCD_D7_PIN, LCD_COLS_NUM, LCD_ROWS_NUM, 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 --

11.13 Stream.cpp File Reference

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

11.14 Tone.cpp File Reference

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

11.15 USBCore.cpp File Reference

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

11.16 WInterrupts.c File Reference

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

11.17 wiring.c File Reference

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

11.18 wiring_analog.c File Reference

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

11.19 wiring_digital.c File Reference

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

11.20 wiring_pulse.c File Reference

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

11.21 wiring_shift.c File Reference

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

11.22 WMath.cpp File Reference

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

11.23 WString.cpp File Reference

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

Index

- ~aDCDisplay
 - aDCDisplay, [27](#)
- ~aDCEngine
 - aDCEngine, [32](#)
- ~aDCSettings
 - aDCSettings, [40](#)
- ~aLCD
 - aLCD, [55](#)
- ~aStepper
 - aStepper, [57](#)
- _adjustLoadCurrent
 - aDCEngine, [32](#)
- _crc8
 - aDCSettings, [41](#)
- _dimBacklight
 - aDCDisplay, [27](#)
- _dimmingBacklight
 - aDCDisplay, [27](#)
- _eepromCalibrationBackup
 - aDCSettings, [41](#)
- _eepromCalibrationRestore
 - aDCSettings, [41](#)
- _eepromCheckForMagicNumbers
 - aDCSettings, [41](#)
- _eepromReset
 - aDCSettings, [41](#)
- _eepromRestore
 - aDCSettings, [42](#)
- _eepromWriteMagicNumbers
 - aDCSettings, [42](#)
- _enableData
 - aDCSettings, [42](#)
- _enableDataCheck
 - aDCSettings, [42](#)
- _getADC
 - aDCEngine, [32](#)
- _getInputVoltage
 - aDCEngine, [32](#)
- _getMeasuredCurrent
 - aDCEngine, [32](#)
- _getSettings
 - aDCEngine, [33](#)
- _getTemp
 - aDCEngine, [33](#)
- _handleButtonEvent
 - aDCEngine, [33](#)
- _handleLoggingAndRemote
 - aDCEngine, [33](#)
- _pow

- aStepper, [58](#)
- _setDAC
 - aDCEngine, [33](#)
- _setValue
 - aDCSettings, [42](#)
- _updateFanSpeed
 - aDCEngine, [34](#)
- _updateLoadCurrent
 - aDCEngine, [34](#)
- _wakeupBacklight
 - aDCDisplay, [27](#)
- aDCSettings
 - CALIBRATION_DAC_CURRENT, [39](#)
 - CALIBRATION_MAX, [39](#)
 - CALIBRATION_READ_CURRENT, [39](#)
 - CALIBRATION_VOLTAGE, [39](#)
 - CALIBRATION_VOLTAGE_DROP, [39](#)
 - DISPLAY_MODE_SETUP, [40](#)
 - DISPLAY_MODE_UNKNOWN, [40](#)
 - DISPLAY_MODE_VALUES, [40](#)
 - OPERATION_MODE_READ, [40](#)
 - OPERATION_MODE_SET, [40](#)
 - OPERATION_MODE_UNKNOWN, [40](#)
 - SELECTION_MODE_CURRENT, [40](#)
 - SELECTION_MODE_POWER, [40](#)
 - SELECTION_MODE_UNKNOWN, [40](#)
 - SETTING_ERROR_OVERSIZED, [40](#)
 - SETTING_ERROR_UNDERSIZED, [40](#)
 - SETTING_ERROR_VALID, [40](#)
- ADC_CHIPSELECT_PIN
 - aDCLoad.h, [71](#)
- ADC_TEMPSENSE1_CHAN
 - aDCLoad.h, [71](#)
- ADC_TEMPSENSE2_CHAN
 - aDCLoad.h, [71](#)
- aDCDisplay, [25](#)
 - ~aDCDisplay, [27](#)
 - _dimBacklight, [27](#)
 - _dimmingBacklight, [27](#)
 - _wakeupBacklight, [27](#)
 - aDCDisplay, [26](#)
 - aDCDisplay, [26](#)
 - aDCEngine, [35](#)
 - isBacklightDimmed, [27](#)
 - m_Parent, [29](#)
 - m_dimmed, [29](#)
 - m_dimmerTick, [29](#)
 - m_nextUpdate, [29](#)
 - pingBacklight, [28](#)

- setup, [28](#)
 - showBanner, [28](#)
 - updateDisplay, [28](#)
 - updateField, [28](#)
- aDCEngine, [29](#)
 - ~aDCEngine, [32](#)
 - _adjustLoadCurrent, [32](#)
 - _getADC, [32](#)
 - _getInputVoltage, [32](#)
 - _getMeasuredCurrent, [32](#)
 - _getSettings, [33](#)
 - _getTemp, [33](#)
 - _handleButtonEvent, [33](#)
 - _handleLoggingAndRemote, [33](#)
 - _setDAC, [33](#)
 - _updateFanSpeed, [34](#)
 - _updateLoadCurrent, [34](#)
- aDCDisplay, [35](#)
- aDCEngine, [31](#)
- aDCEngine, [31](#)
- m_Data, [35](#)
- m_RXbuffer, [35](#)
- m_RXoffset, [35](#)
- m_encoder, [35](#)
- RXBUFFER_MAXLEN, [35](#)
- run, [34](#)
- service, [34](#)
- setup, [34](#)
- aDCLoad.cpp, [61](#)
 - floatRounding, [62](#)
 - getNumericalLength, [62](#)
 - serialAvailable, [64](#)
 - serialFlush, [64](#)
 - serialPrint, [64](#), [65](#)
 - serialPrintln, [65](#)
 - serialRead, [66](#)
 - serialWrite, [66](#)
- aDCLoad.h, [66](#)
 - AUTOLOCK_TIMEOUT, [72](#)
 - BACKLIGHT_TIMEOUT, [72](#)
 - CURRENT_MAXIMUM, [72](#)
 - DAC_CURRENT_CHAN, [72](#)
 - DAC_FAN_CHAN, [72](#)
 - ENCODER_A_PIN, [73](#)
 - ENCODER_B_PIN, [73](#)
 - ENCODER_PB_PIN, [73](#)
 - FEATURE_AUTODIM, [73](#)
 - FEATURE_AUTOLOCK, [74](#)
 - FEATURE_LOCKED, [74](#)
 - FEATURE_LOGGING, [74](#)
 - FEATURE_OCP, [74](#)
 - FEATURE_OTP, [74](#)
 - FEATURE_OVP, [74](#)
 - FEATURE_USB, [75](#)
 - GLYPH_LOCK, [75](#)
 - GLYPH_USB, [75](#)
 - GLYPH_X1, [75](#)
 - GLYPH_X10, [75](#)
 - GLYPH_X100, [75](#)
 - GLYPH_X1000, [75](#)
 - ISRCallback, [71](#)
 - LCD_COLS_NUM, [75](#)
 - LCD_D0_PIN, [76](#)
 - LCD_D1_PIN, [76](#)
 - LCD_D2_PIN, [76](#)
 - LCD_D3_PIN, [76](#)
 - LCD_D4_PIN, [76](#)
 - LCD_D5_PIN, [76](#)
 - LCD_D6_PIN, [76](#)
 - LCD_D7_PIN, [76](#)
 - LCD_ENABLE_PIN, [76](#)
 - LCD_ROWS_NUM, [76](#)
 - LCD_RS_PIN, [76](#)
 - LOGGING_RATE, [77](#)
 - MAX_POWER, [71](#)
 - OFFSET_TEMP, [77](#)
 - OFFSET_UNIT, [77](#)
 - OFFSET_VALUE, [77](#)
 - POWER_MAXIMUM, [78](#)
 - VOLTAGE_MAXIMUM, [78](#)
- aDCSettings, [35](#)
 - ~aDCSettings, [40](#)
 - _crc8, [41](#)
 - _eepromCalibrationBackup, [41](#)
 - _eepromCalibrationRestore, [41](#)
 - _eepromCheckForMagicNumbers, [41](#)
 - _eepromReset, [41](#)
 - _eepromRestore, [42](#)
 - _eepromWriteMagicNumbers, [42](#)
 - _enableData, [42](#)
 - _enableDataCheck, [42](#)
 - _setValue, [42](#)
- aDCSettings, [40](#)
- aDCSettings, [40](#)
- backupCalibration, [43](#)
- CalibrationValues_t, [39](#)
- DATA_CURRENT_READ, [51](#)
- DATA_CURRENT_SETS, [51](#)
- DATA_DISPLAY, [51](#)
- DATA_ENCODER, [51](#)
- DATA_OPERATION, [51](#)
- DATA_POWER_READ, [51](#)
- DATA_POWER_SETS, [52](#)
- DATA_SELECTION, [52](#)
- DATA_TEMPERATURE, [52](#)
- DATA_VOLTAGE, [52](#)
- DisplayMode_t, [39](#)
- enableAlarm, [43](#)
- enableFeature, [43](#)
- getCalibrationMode, [43](#)
- getCalibrationValues, [43](#)
- getCurrent, [44](#)
- getCurrentDAC, [44](#)
- getDisplayMode, [44](#)
- getEncoderPosition, [44](#)
- getFanSpeed, [44](#)

- getOperationMode, 44
- getPower, 45
- getPrevNextMode, 45
- getSelectionMode, 45
- getTemperature, 45
- getVoltage, 45
- incEncoderPosition, 45
- isAutolocked, 46
- isDataEnabled, 46
- isFeatureEnabled, 46
- m_calibrationValues, 52
- m_currentDAC, 52
- m_dats, 52
- m_dispMode, 52
- m_encoderPos, 52
- m_fanSpeed, 52
- m_features, 52
- m_lockTick, 53
- m_mode, 53
- m_operationMode, 53
- m_operationTick, 53
- m_readCurrent, 53
- m_readPower, 53
- m_readTemperature, 53
- m_readVoltage, 53
- m_setsCurrent, 53
- m_setsPower, 53
- OperationMode_t, 40
- pingAutolock, 46
- pingOperationMode, 46
- restoreCalibration, 47
- SelectionMode_t, 40
- setCalibrationMode, 47
- setCalibrationValues, 47
- setCurrent, 47
- setCurrentDAC, 47
- setDisplayMode, 48
- setEncoderPosition, 48
- setFanSpeed, 48
- setOperationMode, 48
- setPower, 48
- setSelectionMode, 50
- setTemperature, 50
- setVoltage, 50
- SettingError_t, 40
- syncData, 50
- updateOperationMode, 51
- updateValuesFromMode, 51
- aDCSettings::_eepromCalibrationValue_t, 25
 - c, 25
 - v, 25
- aDCSettings::CalibrationData_t, 59
 - offset, 60
 - slope, 60
- ALARM_OC_X_COORD
 - aDCLoad.h, 71
- ALARM_OC_Y_COORD
 - aDCLoad.h, 71
- ALARM_OT_X_COORD
 - aDCLoad.h, 71
- ALARM_OT_Y_COORD
 - aDCLoad.h, 72
- ALARM_OV_X_COORD
 - aDCLoad.h, 72
- ALARM_OV_Y_COORD
 - aDCLoad.h, 72
- aLCD, 54
 - ~aLCD, 55
 - aLCD, 54
 - aLCD, 54
 - begin, 55
 - clearValue, 55
 - m_cols, 56
 - m_curCol, 56
 - m_curRow, 56
 - m_rows, 56
 - printCenter, 55, 56
 - setCursor, 56
- aStepper, 56
 - ~aStepper, 57
 - _pow, 58
 - aStepper, 57
 - aStepper, 57
 - getMult, 58
 - getValue, 58
 - getValueFromMode, 58
 - increment, 58
 - isSynced, 58
 - m_inc, 59
 - m_incPrev, 59
 - MAX_VALUE, 59
 - reset, 59
 - sync, 59
- AUTOLOCK_TIMEOUT
 - aDCLoad.h, 72
- BACKLIGHT_TIMEOUT
 - aDCLoad.h, 72
- backupCalibration
 - aDCSettings, 43
- begin
 - aLCD, 55
- c
 - aDCSettings::_eepromCalibrationValue_t, 25
- CALIBRATION_DAC_CURRENT
 - aDCSettings, 39
- CALIBRATION_MAX
 - aDCSettings, 39
- CALIBRATION_READ_CURRENT
 - aDCSettings, 39
- CALIBRATION_VOLTAGE
 - aDCSettings, 39
- CALIBRATION_VOLTAGE_DROP
 - aDCSettings, 39
- CDC.cpp, 78
- CURRENT_MAXIMUM

- aDCLoad.h, [72](#)
- CalibrationValues_t
 - aDCSettings, [39](#)
- clearValue
 - aLCD, [55](#)
- DISPLAY_MODE_SETUP
 - aDCSettings, [40](#)
- DISPLAY_MODE_UNKNOWN
 - aDCSettings, [40](#)
- DISPLAY_MODE_VALUES
 - aDCSettings, [40](#)
- DAC_CHIPSELECT_PIN
 - aDCLoad.h, [72](#)
- DAC_CURRENT_CHAN
 - aDCLoad.h, [72](#)
- DAC_FAN_CHAN
 - aDCLoad.h, [72](#)
- DATA_CURRENT_READ
 - aDCSettings, [51](#)
- DATA_CURRENT_SETS
 - aDCSettings, [51](#)
- DATA_DISPLAY
 - aDCSettings, [51](#)
- DATA_ENCODER
 - aDCSettings, [51](#)
- DATA_OPERATION
 - aDCSettings, [51](#)
- DATA_POWER_READ
 - aDCSettings, [51](#)
- DATA_POWER_SETS
 - aDCSettings, [52](#)
- DATA_SELECTION
 - aDCSettings, [52](#)
- DATA_TEMPERATURE
 - aDCSettings, [52](#)
- DATA_VOLTAGE
 - aDCSettings, [52](#)
- DisplayMode_t
 - aDCSettings, [39](#)
- EEPROM_ADDR_MAGIC
 - aDCLoad.h, [73](#)
- ENCODER_A_PIN
 - aDCLoad.h, [73](#)
- ENCODER_B_PIN
 - aDCLoad.h, [73](#)
- ENCODER_PB_PIN
 - aDCLoad.h, [73](#)
- enableAlarm
 - aDCSettings, [43](#)
- enableFeature
 - aDCSettings, [43](#)
- engine
 - sketch.cpp, [80](#)
- FEATURE_AUTODIM
 - aDCLoad.h, [73](#)
- FEATURE_AUTOLOCK
 - aDCLoad.h, [74](#)
- FEATURE_LOCKED
 - aDCLoad.h, [74](#)
- FEATURE_LOGGING
 - aDCLoad.h, [74](#)
- FEATURE_OCP
 - aDCLoad.h, [74](#)
- FEATURE_OTP
 - aDCLoad.h, [74](#)
- FEATURE_OVP
 - aDCLoad.h, [74](#)
- FEATURE_USB
 - aDCLoad.h, [75](#)
- floatRounding
 - aDCLoad.cpp, [62](#)
- GLYPH_LOCK
 - aDCLoad.h, [75](#)
- GLYPH_USB
 - aDCLoad.h, [75](#)
- GLYPH_X1
 - aDCLoad.h, [75](#)
- GLYPH_X10
 - aDCLoad.h, [75](#)
- GLYPH_X100
 - aDCLoad.h, [75](#)
- GLYPH_X1000
 - aDCLoad.h, [75](#)
- getCalibrationMode
 - aDCSettings, [43](#)
- getCalibrationValues
 - aDCSettings, [43](#)
- getCurrent
 - aDCSettings, [44](#)
- getCurrentDAC
 - aDCSettings, [44](#)
- getDisplayMode
 - aDCSettings, [44](#)
- getEncoderPosition
 - aDCSettings, [44](#)
- getFanSpeed
 - aDCSettings, [44](#)
- getMult
 - aStepper, [58](#)
- getNumericalLength
 - aDCLoad.cpp, [62](#)
- getOperationMode
 - aDCSettings, [44](#)
- getPower
 - aDCSettings, [45](#)
- getPrevNextMode
 - aDCSettings, [45](#)
- getSelectionMode
 - aDCSettings, [45](#)
- getTemperature
 - aDCSettings, [45](#)
- getValue
 - aStepper, [58](#)
- getValueFromMode

- aStepper, 58
- getVoltage
 - aDCSettings, 45
- HID.cpp, 78
- HardwareSerial.cpp, 78
- IPAddress.cpp, 79
- ISRCallback
 - aDCLoad.h, 71
- incEncoderPosition
 - aDCSettings, 45
- increment
 - aStepper, 58
- isAutolocked
 - aDCSettings, 46
- isBacklightDimmed
 - aDCDisplay, 27
- isDataEnabled
 - aDCSettings, 46
- isFeatureEnabled
 - aDCSettings, 46
- isSynced
 - aStepper, 58
- isr
 - sketch.cpp, 79
- LCD_COLS_NUM
 - aDCLoad.h, 75
- LCD_D0_PIN
 - aDCLoad.h, 76
- LCD_D1_PIN
 - aDCLoad.h, 76
- LCD_D2_PIN
 - aDCLoad.h, 76
- LCD_D3_PIN
 - aDCLoad.h, 76
- LCD_D4_PIN
 - aDCLoad.h, 76
- LCD_D5_PIN
 - aDCLoad.h, 76
- LCD_D6_PIN
 - aDCLoad.h, 76
- LCD_D7_PIN
 - aDCLoad.h, 76
- LCD_ENABLE_PIN
 - aDCLoad.h, 76
- LCD_ROWS_NUM
 - aDCLoad.h, 76
- LCD_RS_PIN
 - aDCLoad.h, 76
- LED_BACKLIGHT_PIN
 - aDCLoad.h, 76
- LOCK_ICON_X_COORD
 - aDCLoad.h, 77
- LOCK_ICON_Y_COORD
 - aDCLoad.h, 77
- LOGGING_RATE
 - aDCLoad.h, 77
- libraries.cpp, 79
- loop
 - sketch.cpp, 79
- m_Data
 - aDCEngine, 35
- m_Parent
 - aDCDisplay, 29
- m_RXbuffer
 - aDCEngine, 35
- m_RXoffset
 - aDCEngine, 35
- m_calibrationValues
 - aDCSettings, 52
- m_cols
 - aLCD, 56
- m_curCol
 - aLCD, 56
- m_curRow
 - aLCD, 56
- m_currentDAC
 - aDCSettings, 52
- m_datas
 - aDCSettings, 52
- m_dimmed
 - aDCDisplay, 29
- m_dimmerTick
 - aDCDisplay, 29
- m_dispMode
 - aDCSettings, 52
- m_encoder
 - aDCEngine, 35
- m_encoderPos
 - aDCSettings, 52
- m_fanSpeed
 - aDCSettings, 52
- m_features
 - aDCSettings, 52
- m_inc
 - aStepper, 59
- m_incPrev
 - aStepper, 59
- m_lockTick
 - aDCSettings, 53
- m_mode
 - aDCSettings, 53
- m_nextUpdate
 - aDCDisplay, 29
- m_operationMode
 - aDCSettings, 53
- m_operationTick
 - aDCSettings, 53
- m_readCurrent
 - aDCSettings, 53
- m_readPower
 - aDCSettings, 53
- m_readTemperature
 - aDCSettings, 53
- m_readVoltage

- aDCSettings, [53](#)
- m_rows
 - aLCD, [56](#)
- m_setsCurrent
 - aDCSettings, [53](#)
- m_setsPower
 - aDCSettings, [53](#)
- MAX_POWER
 - aDCLoad.h, [71](#)
- MAX_VALUE
 - aStepper, [59](#)
- main.cpp, [79](#)
- new.cpp, [79](#)
- OPERATION_MODE_READ
 - aDCSettings, [40](#)
- OPERATION_MODE_SET
 - aDCSettings, [40](#)
- OPERATION_MODE_UNKNOWN
 - aDCSettings, [40](#)
- OFFSET_MARKER_LEFT
 - aDCLoad.h, [77](#)
- OFFSET_TEMP
 - aDCLoad.h, [77](#)
- OFFSET_UNIT
 - aDCLoad.h, [77](#)
- OFFSET_VALUE
 - aDCLoad.h, [77](#)
- offset
 - aDCSettings::CalibrationData_t, [60](#)
- OperationMode_t
 - aDCSettings, [40](#)
- POWER_MAXIMUM
 - aDCLoad.h, [78](#)
- pingAutolock
 - aDCSettings, [46](#)
- pingBacklight
 - aDCDisplay, [28](#)
- pingOperationMode
 - aDCSettings, [46](#)
- Print.cpp, [79](#)
- printCenter
 - aLCD, [55, 56](#)
- README.md, [79](#)
- RXBUFFER_MAXLEN
 - aDCEngine, [35](#)
- reset
 - aStepper, [59](#)
- restoreCalibration
 - aDCSettings, [47](#)
- run
 - aDCEngine, [34](#)
- SELECTION_MODE_CURRENT
 - aDCSettings, [40](#)
- SELECTION_MODE_POWER
 - aDCSettings, [40](#)
- SELECTION_MODE_UNKNOWN
 - aDCSettings, [40](#)
- SETTING_ERROR_OVERSIZED
 - aDCSettings, [40](#)
- SETTING_ERROR_UNDERSIZED
 - aDCSettings, [40](#)
- SETTING_ERROR_VALID
 - aDCSettings, [40](#)
- SelectionMode_t
 - aDCSettings, [40](#)
- serialAvailable
 - aDCLoad.cpp, [64](#)
- serialFlush
 - aDCLoad.cpp, [64](#)
- serialPrint
 - aDCLoad.cpp, [64, 65](#)
- serialPrintln
 - aDCLoad.cpp, [65](#)
- serialRead
 - aDCLoad.cpp, [66](#)
- serialWrite
 - aDCLoad.cpp, [66](#)
- service
 - aDCEngine, [34](#)
- setCalibrationMode
 - aDCSettings, [47](#)
- setCalibrationValues
 - aDCSettings, [47](#)
- setCurrent
 - aDCSettings, [47](#)
- setCurrentDAC
 - aDCSettings, [47](#)
- setCursor
 - aLCD, [56](#)
- setDisplayMode
 - aDCSettings, [48](#)
- setEncoderPosition
 - aDCSettings, [48](#)
- setFanSpeed
 - aDCSettings, [48](#)
- setOperationMode
 - aDCSettings, [48](#)
- setPower
 - aDCSettings, [48](#)
- setSelectionMode
 - aDCSettings, [50](#)
- setTemperature
 - aDCSettings, [50](#)
- setVoltage
 - aDCSettings, [50](#)
- SettingError_t
 - aDCSettings, [40](#)
- setup
 - aDCDisplay, [28](#)
 - aDCEngine, [34](#)
 - sketch.cpp, [80](#)
- showBanner

- aDCDisplay, [28](#)
- sketch.cpp, [79](#)
 - engine, [80](#)
 - isr, [79](#)
 - loop, [79](#)
 - setup, [80](#)
- slope
 - aDCSettings::CalibrationData_t, [60](#)
- Stream.cpp, [80](#)
- sync
 - aStepper, [59](#)
- syncData
 - aDCSettings, [50](#)
- TEMPERATURE_MAXIMUM
 - aDCLoad.h, [78](#)
- Tone.cpp, [80](#)
- USB_ICON_X_COORD
 - aDCLoad.h, [78](#)
- USB_ICON_Y_COORD
 - aDCLoad.h, [78](#)
- USBCore.cpp, [80](#)
- updateDisplay
 - aDCDisplay, [28](#)
- updateField
 - aDCDisplay, [28](#)
- updateOperationMode
 - aDCSettings, [51](#)
- updateValuesFromMode
 - aDCSettings, [51](#)
- v
 - aDCSettings::_eepromCalibrationValue_t, [25](#)
- VOLTAGE_MAXIMUM
 - aDCLoad.h, [78](#)
- WInterrupts.c, [80](#)
- WMath.cpp, [81](#)
- WString.cpp, [81](#)
- wiring.c, [80](#)
- wiring_analog.c, [80](#)
- wiring_digital.c, [80](#)
- wiring_pulse.c, [80](#)
- wiring_shift.c, [81](#)