



**Manual for aDCLoad**  
version 2.6

Written by F1RMB, Daniel Caujolle-Bert ©2014-2015



# Contents

<b>1</b>	<b>Arduino Programmable Constant Current Power Resistance Load</b>	<b>1</b>
1.1	Original README.md from Lee Wiggins . . . . .	1
1.2	Informations on this code from Daniel Caujolle-Bert . . . . .	1
<b>2</b>	<b>ATmega32U4 fuses settings</b>	<b>3</b>
<b>3</b>	<b>Hardware modifications for version 2.6</b>	<b>5</b>
<b>4</b>	<b>User interface overview</b>	<b>7</b>
4.1	Extra control since v2.6 . . . . .	9
<b>5</b>	<b>Remote Commands</b>	<b>11</b>
5.1	Get Identification . . . . .	11
5.2	Current setting getter . . . . .	11
5.3	Current setting setter . . . . .	11
5.4	Calibration values getter . . . . .	12
5.5	Calibration . . . . .	12
5.6	DAC value setter (calibration purpose) . . . . .	12
5.7	Current readed getter . . . . .	12
5.8	Voltage readed getter . . . . .	13
5.9	Logging enability . . . . .	13
5.10	Logging enability . . . . .	13
5.11	Pulse value getter . . . . .	13
5.12	Pulse value setter . . . . .	13
5.13	Input Relay status getter . . . . .	14
5.14	Input Relay status setter . . . . .	14
5.15	Return value . . . . .	14
<b>6</b>	<b>Logging data format</b>	<b>15</b>
6.1	CSV logging format . . . . .	15
<b>7</b>	<b>Calibration Process</b>	<b>17</b>
<b>8</b>	<b>Hierarchical Index</b>	<b>21</b>

8.1	Class Hierarchy	21
<b>9</b>	<b>Class Index</b>	<b>23</b>
9.1	Class List	23
<b>10</b>	<b>File Index</b>	<b>25</b>
10.1	File List	25
<b>11</b>	<b>Class Documentation</b>	<b>27</b>
11.1	aDCSettings::_eepromCalibrationValue_t Union Reference	27
11.1.1	Detailed Description	27
11.1.2	Member Data Documentation	27
11.1.2.1	c	27
11.1.2.2	v	27
11.2	aDCDisplay Class Reference	27
11.2.1	Detailed Description	28
11.2.2	Constructor & Destructor Documentation	28
11.2.2.1	aDCDisplay	28
11.2.2.2	~aDCDisplay	29
11.2.3	Member Function Documentation	29
11.2.3.1	_dimBacklight	29
11.2.3.2	_dimmingBacklight	29
11.2.3.3	_wakeupBacklight	29
11.2.3.4	isBacklightDimmed	29
11.2.3.5	pingBacklight	30
11.2.3.6	setup	30
11.2.3.7	showBanner	30
11.2.3.8	updateDisplay	30
11.2.3.9	updateField	30
11.2.4	Member Data Documentation	30
11.2.4.1	m_dimmed	30
11.2.4.2	m_dimmerTick	31
11.2.4.3	m_nextUpdate	31
11.2.4.4	m_Parent	31
11.3	aDCEngine Class Reference	31
11.3.1	Detailed Description	32
11.3.2	Constructor & Destructor Documentation	32
11.3.2.1	aDCEngine	32
11.3.2.2	~aDCEngine	33
11.3.3	Member Function Documentation	33
11.3.3.1	_adjustLoadCurrent	33

11.3.3.2	<a href="#">_getADC</a>	33
11.3.3.3	<a href="#">_getInputVoltage</a>	33
11.3.3.4	<a href="#">_getMeasuredCurrent</a>	33
11.3.3.5	<a href="#">_getTemp</a>	34
11.3.3.6	<a href="#">_handleButtonEvent</a>	34
11.3.3.7	<a href="#">_handleLoggingAndRemote</a>	34
11.3.3.8	<a href="#">_setDAC</a>	34
11.3.3.9	<a href="#">_updateFanSpeed</a>	34
11.3.3.10	<a href="#">_updateLoadCurrent</a>	34
11.3.3.11	<a href="#">getEncoder</a>	35
11.3.3.12	<a href="#">getSettings</a>	35
11.3.3.13	<a href="#">run</a>	35
11.3.3.14	<a href="#">setInput</a>	35
11.3.3.15	<a href="#">setup</a>	35
11.3.4	<a href="#">Friends And Related Function Documentation</a>	35
11.3.4.1	<a href="#">aDCDisplay</a>	35
11.3.5	<a href="#">Member Data Documentation</a>	35
11.3.5.1	<a href="#">m_Data</a>	35
11.3.5.2	<a href="#">m_encoder</a>	35
11.3.5.3	<a href="#">m_inputRelay</a>	35
11.3.5.4	<a href="#">m_RXbuffer</a>	35
11.3.5.5	<a href="#">m_RXoffset</a>	36
11.3.5.6	<a href="#">RXBUFFER_MAXLEN</a>	36
11.4	<a href="#">aDCSettings Class Reference</a>	36
11.4.1	<a href="#">Detailed Description</a>	40
11.4.2	<a href="#">Member Enumeration Documentation</a>	40
11.4.2.1	<a href="#">CalibrationValues_t</a>	40
11.4.2.2	<a href="#">DisplayMode_t</a>	41
11.4.2.3	<a href="#">OperationMode_t</a>	41
11.4.2.4	<a href="#">SelectionMode_t</a>	41
11.4.2.5	<a href="#">SettingError_t</a>	41
11.4.3	<a href="#">Constructor &amp; Destructor Documentation</a>	41
11.4.3.1	<a href="#">aDCSettings</a>	41
11.4.3.2	<a href="#">~aDCSettings</a>	42
11.4.4	<a href="#">Member Function Documentation</a>	42
11.4.4.1	<a href="#">_crc8</a>	42
11.4.4.2	<a href="#">_eepromCalibrationBackup</a>	42
11.4.4.3	<a href="#">_eepromCalibrationRestore</a>	42
11.4.4.4	<a href="#">_eepromCheckForMagicNumbers</a>	42
11.4.4.5	<a href="#">_eepromReset</a>	43

11.4.4.6	<a href="#">_eepromRestore</a>	43
11.4.4.7	<a href="#">_eepromWriteMagicNumbers</a>	43
11.4.4.8	<a href="#">_enableData</a>	43
11.4.4.9	<a href="#">_enableDataCheck</a>	43
11.4.4.10	<a href="#">_setValue</a>	44
11.4.4.11	<a href="#">backupCalibration</a>	44
11.4.4.12	<a href="#">enableAlarm</a>	44
11.4.4.13	<a href="#">enableFeature</a>	44
11.4.4.14	<a href="#">enablePulse</a>	45
11.4.4.15	<a href="#">getCalibrationMode</a>	45
11.4.4.16	<a href="#">getCalibrationValues</a>	45
11.4.4.17	<a href="#">getCurrent</a>	45
11.4.4.18	<a href="#">getCurrentDAC</a>	46
11.4.4.19	<a href="#">getDisplayMode</a>	46
11.4.4.20	<a href="#">getEncoderPosition</a>	46
11.4.4.21	<a href="#">getFanSpeed</a>	46
11.4.4.22	<a href="#">getOperationMode</a>	46
11.4.4.23	<a href="#">getPower</a>	46
11.4.4.24	<a href="#">getPrevNextMode</a>	46
11.4.4.25	<a href="#">getPulse</a>	47
11.4.4.26	<a href="#">getSelectionMode</a>	47
11.4.4.27	<a href="#">getTemperature</a>	47
11.4.4.28	<a href="#">getVoltage</a>	47
11.4.4.29	<a href="#">incEncoderPosition</a>	47
11.4.4.30	<a href="#">isAutolocked</a>	48
11.4.4.31	<a href="#">isDataEnabled</a>	48
11.4.4.32	<a href="#">isFeatureEnabled</a>	48
11.4.4.33	<a href="#">isPulseEnabled</a>	48
11.4.4.34	<a href="#">isPulseHigh</a>	48
11.4.4.35	<a href="#">pingAutolock</a>	48
11.4.4.36	<a href="#">pingOperationMode</a>	49
11.4.4.37	<a href="#">restoreCalibration</a>	49
11.4.4.38	<a href="#">setCalibationMode</a>	49
11.4.4.39	<a href="#">setCalibrationValues</a>	49
11.4.4.40	<a href="#">setCurrent</a>	49
11.4.4.41	<a href="#">setCurrentDAC</a>	49
11.4.4.42	<a href="#">setDisplayMode</a>	50
11.4.4.43	<a href="#">setEncoderPosition</a>	50
11.4.4.44	<a href="#">setFanSpeed</a>	50
11.4.4.45	<a href="#">setOperationMode</a>	50

11.4.4.46	setPower	51
11.4.4.47	setPulse	52
11.4.4.48	setPulseHigh	52
11.4.4.49	setSelectionMode	52
11.4.4.50	setTemperature	52
11.4.4.51	setVoltage	53
11.4.4.52	syncData	53
11.4.4.53	updateOperationMode	53
11.4.4.54	updateValuesFromMode	53
11.4.5	Member Data Documentation	54
11.4.5.1	DATA_CURRENT_READ	54
11.4.5.2	DATA_CURRENT_SETS	54
11.4.5.3	DATA_DISPLAY	54
11.4.5.4	DATA_ENCODER	54
11.4.5.5	DATA_IN_CALIBRATION	54
11.4.5.6	DATA_OPERATION	54
11.4.5.7	DATA_POWER_READ	54
11.4.5.8	DATA_POWER_SETS	54
11.4.5.9	DATA_PULSE_SETS	54
11.4.5.10	DATA_SELECTION	54
11.4.5.11	DATA_TEMPERATURE	54
11.4.5.12	DATA_VOLTAGE	54
11.4.5.13	m_calibrationValues	55
11.4.5.14	m_currentDAC	55
11.4.5.15	m_datas	55
11.4.5.16	m_dispMode	55
11.4.5.17	m_encoderPos	55
11.4.5.18	m_fanSpeed	55
11.4.5.19	m_features	55
11.4.5.20	m_lockTick	55
11.4.5.21	m_mode	55
11.4.5.22	m_operationMode	55
11.4.5.23	m_operationTick	56
11.4.5.24	m_Parent	56
11.4.5.25	m_pulseEnabled	56
11.4.5.26	m_pulseHigh	56
11.4.5.27	m_readCurrent	56
11.4.5.28	m_readPower	56
11.4.5.29	m_readTemperature	56
11.4.5.30	m_readVoltage	56

11.4.5.31	<code>m_setsCurrent</code>	56
11.4.5.32	<code>m_setsPower</code>	56
11.4.5.33	<code>m_setsPulse</code>	56
11.5	<code>aInputRelay</code> Class Reference	56
11.5.1	Detailed Description	57
11.5.2	Constructor & Destructor Documentation	57
11.5.2.1	<code>aInputRelay</code>	57
11.5.2.2	<code>~aInputRelay</code>	57
11.5.3	Member Function Documentation	57
11.5.3.1	<code>isClicked</code>	57
11.5.3.2	<code>isInput</code>	58
11.5.3.3	<code>service</code>	58
11.5.3.4	<code>setInput</code>	58
11.5.4	Member Data Documentation	58
11.5.4.1	<code>m_btnPin</code>	58
11.5.4.2	<code>m_btnState</code>	58
11.5.4.3	<code>m_clicked</code>	58
11.5.4.4	<code>m_isON</code>	58
11.5.4.5	<code>m_relayPin</code>	58
11.6	<code>aLCD</code> Class Reference	58
11.6.1	Detailed Description	59
11.6.2	Constructor & Destructor Documentation	59
11.6.2.1	<code>aLCD</code>	59
11.6.2.2	<code>~aLCD</code>	59
11.6.3	Member Function Documentation	60
11.6.3.1	<code>begin</code>	60
11.6.3.2	<code>clearValue</code>	60
11.6.3.3	<code>printCenter</code>	60
11.6.3.4	<code>printCenter</code>	60
11.6.3.5	<code>setCursor</code>	60
11.6.4	Member Data Documentation	61
11.6.4.1	<code>m_cols</code>	61
11.6.4.2	<code>m_curCol</code>	61
11.6.4.3	<code>m_curRow</code>	61
11.6.4.4	<code>m_rows</code>	61
11.7	<code>aStepper</code> Class Reference	61
11.7.1	Detailed Description	62
11.7.2	Constructor & Destructor Documentation	62
11.7.2.1	<code>aStepper</code>	62
11.7.2.2	<code>~aStepper</code>	62



11.7.3	Member Function Documentation	62
11.7.3.1	_pow	62
11.7.3.2	getMult	63
11.7.3.3	getValue	63
11.7.3.4	getValueFromMode	63
11.7.3.5	increment	63
11.7.3.6	isSynced	63
11.7.3.7	reset	63
11.7.3.8	sync	64
11.7.4	Member Data Documentation	64
11.7.4.1	m_inc	64
11.7.4.2	m_incPrev	64
11.7.4.3	MAX_VALUE	64
11.8	aDCSettings::CalibrationData_t Struct Reference	64
11.8.1	Detailed Description	64
11.8.2	Member Data Documentation	64
11.8.2.1	offset	64
11.8.2.2	slope	65
<b>12</b>	<b>File Documentation</b>	<b>67</b>
12.1	aDCLoad.cpp File Reference	67
12.1.1	Detailed Description	68
12.1.2	Function Documentation	68
12.1.2.1	floatRounding	68
12.1.2.2	getNumericalLength	69
12.1.2.3	serialAvailable	69
12.1.2.4	serialFlush	69
12.1.2.5	serialPrint	69
12.1.2.6	serialPrint	69
12.1.2.7	serialPrint	70
12.1.2.8	serialPrint	70
12.1.2.9	serialPrint	70
12.1.2.10	serialPrintln	70
12.1.2.11	serialPrintln	70
12.1.2.12	serialPrintln	71
12.1.2.13	serialRead	71
12.1.2.14	serialWrite	71
12.1.2.15	timer1ISR	71
12.1.2.16	timer3ISR	71
12.1.3	Variable Documentation	72

12.1.3.1	pThis	72
12.2	aDCLoad.h File Reference	72
12.2.1	Detailed Description	76
12.2.2	Macro Definition Documentation	76
12.2.2.1	HAS_INPUT_RELAY	76
12.2.2.2	HAS_TWIN_MOSFET	76
12.2.2.3	MAX_POWER	76
12.2.3	Variable Documentation	77
12.2.3.1	ADC_CHIPSELECT_PIN	77
12.2.3.2	ADC_INPUTVOLTAGE_CHAN	77
12.2.3.3	ADC_MEASUREDCURRENT_CHAN	77
12.2.3.4	ADC_TEMPSENSE1_CHAN	77
12.2.3.5	ADC_TEMPSENSE2_CHAN	77
12.2.3.6	ALARM_OC_X_COORD	77
12.2.3.7	ALARM_OC_Y_COORD	77
12.2.3.8	ALARM_OT_X_COORD	77
12.2.3.9	ALARM_OT_Y_COORD	77
12.2.3.10	ALARM_OV_X_COORD	77
12.2.3.11	ALARM_OV_Y_COORD	77
12.2.3.12	AUTOLOCK_TIMEOUT	78
12.2.3.13	BACKLIGHT_TIMEOUT	78
12.2.3.14	CURRENT_MAXIMUM	78
12.2.3.15	DAC_CHIPSELECT_PIN	78
12.2.3.16	DAC_CURRENT_CHAN	78
12.2.3.17	DAC_FAN_CHAN	78
12.2.3.18	DISPLAY_UPDATE_RATE	78
12.2.3.19	EEPROM_ADDR_AUTODIM	78
12.2.3.20	EEPROM_ADDR_AUTOLOCK	78
12.2.3.21	EEPROM_ADDR_CALIBRATION_DAC_CURRENT	78
12.2.3.22	EEPROM_ADDR_CALIBRATION_READ_CURRENT	78
12.2.3.23	EEPROM_ADDR_CALIBRATION_VOLTAGE	78
12.2.3.24	EEPROM_ADDR_CALIBRATION_VOLTAGE_DROP	79
12.2.3.25	EEPROM_ADDR_MAGIC	79
12.2.3.26	EEPROM_CALIBRATION_SIZE	79
12.2.3.27	ENCODER_A_PIN	79
12.2.3.28	ENCODER_B_PIN	79
12.2.3.29	ENCODER_PB_PIN	79
12.2.3.30	ENCODER_STEPS_PER_NOTCH	79
12.2.3.31	FEATURE_AUTODIM	79
12.2.3.32	FEATURE_AUTODIM_VISIBLE	79

12.2.3.33 FEATURE_AUTOLOCK . . . . .	79
12.2.3.34 FEATURE_AUTOLOCK_VISIBLE . . . . .	79
12.2.3.35 FEATURE_LOCKED . . . . .	79
12.2.3.36 FEATURE_LOCKED_VISIBLE . . . . .	80
12.2.3.37 FEATURE_LOGGING . . . . .	80
12.2.3.38 FEATURE_LOGGING_VISIBLE . . . . .	80
12.2.3.39 FEATURE_OCP . . . . .	80
12.2.3.40 FEATURE_OCP_VISIBLE . . . . .	80
12.2.3.41 FEATURE_OTP . . . . .	80
12.2.3.42 FEATURE_OTP_VISIBLE . . . . .	80
12.2.3.43 FEATURE_OVP . . . . .	80
12.2.3.44 FEATURE_OVP_VISIBLE . . . . .	80
12.2.3.45 FEATURE_USB . . . . .	80
12.2.3.46 FEATURE_USB_VISIBLE . . . . .	80
12.2.3.47 GLYPH_CHECKBOX_TICKED . . . . .	80
12.2.3.48 GLYPH_CHECKBOX_UNTICKED . . . . .	81
12.2.3.49 GLYPH_LOCK . . . . .	81
12.2.3.50 GLYPH_USB . . . . .	81
12.2.3.51 GLYPH_X1 . . . . .	81
12.2.3.52 GLYPH_X10 . . . . .	81
12.2.3.53 GLYPH_X100 . . . . .	81
12.2.3.54 GLYPH_X1000 . . . . .	81
12.2.3.55 INPUT_RELAY_BUTTON_PIN . . . . .	81
12.2.3.56 INPUT_RELAY_RELAY_PIN . . . . .	81
12.2.3.57 LCD_COLS_NUM . . . . .	81
12.2.3.58 LCD_D4_PIN . . . . .	81
12.2.3.59 LCD_D5_PIN . . . . .	82
12.2.3.60 LCD_D6_PIN . . . . .	82
12.2.3.61 LCD_D7_PIN . . . . .	82
12.2.3.62 LCD_ENABLE_PIN . . . . .	82
12.2.3.63 LCD_ROWS_NUM . . . . .	82
12.2.3.64 LCD_RS_PIN . . . . .	82
12.2.3.65 LED_BACKLIGHT_PIN . . . . .	82
12.2.3.66 LOCK_ICON_X_COORD . . . . .	82
12.2.3.67 LOCK_ICON_Y_COORD . . . . .	82
12.2.3.68 LOGGING_ICON_X_COORD . . . . .	82
12.2.3.69 LOGGING_ICON_Y_COORD . . . . .	83
12.2.3.70 LOGGING_RATE . . . . .	83
12.2.3.71 OFFSET_MARKER_LEFT . . . . .	83
12.2.3.72 OFFSET_MARKER_RIGHT . . . . .	83

12.2.3.73 OFFSET_SETUP_MARKER_LEFT . . . . .	83
12.2.3.74 OFFSET_SETUP_MARKER_RIGHT . . . . .	83
12.2.3.75 OFFSET_TEMP . . . . .	83
12.2.3.76 OFFSET_UNIT . . . . .	83
12.2.3.77 OFFSET_VALUE . . . . .	83
12.2.3.78 OPERATION_SET_TIMEOUT . . . . .	83
12.2.3.79 POWER_MAXIMUM . . . . .	83
12.2.3.80 PULSE_MAXIMUM . . . . .	83
12.2.3.81 SOFTWARE_VERSION_MAJOR . . . . .	84
12.2.3.82 SOFTWARE_VERSION_MINOR . . . . .	84
12.2.3.83 TEMPERATURE_MAXIMUM . . . . .	84
12.2.3.84 USB_ICON_X_COORD . . . . .	84
12.2.3.85 USB_ICON_Y_COORD . . . . .	84
12.2.3.86 VOLTAGE_MAXIMUM . . . . .	84
12.3 CDC.cpp File Reference . . . . .	84
12.4 HardwareSerial.cpp File Reference . . . . .	84
12.5 HID.cpp File Reference . . . . .	84
12.6 IPAddress.cpp File Reference . . . . .	84
12.7 libraries.cpp File Reference . . . . .	84
12.8 main.cpp File Reference . . . . .	85
12.9 new.cpp File Reference . . . . .	85
12.10Print.cpp File Reference . . . . .	85
12.11sketch.cpp File Reference . . . . .	85
12.11.1 Function Documentation . . . . .	85
12.11.1.1 loop . . . . .	85
12.11.1.2 setup . . . . .	85
12.11.2 Variable Documentation . . . . .	85
12.11.2.1 engine . . . . .	85
12.12Stream.cpp File Reference . . . . .	85
12.13Tone.cpp File Reference . . . . .	86
12.14USBCore.cpp File Reference . . . . .	86
12.15WInterrupts.c File Reference . . . . .	86
12.16wiring.c File Reference . . . . .	86
12.17wiring_analog.c File Reference . . . . .	86
12.18wiring_digital.c File Reference . . . . .	86
12.19wiring_pulse.c File Reference . . . . .	86
12.20wiring_shift.c File Reference . . . . .	86
12.21WMath.cpp File Reference . . . . .	86
12.22WString.cpp File Reference . . . . .	86





## Chapter 1

# Arduino Programmable Constant Current Power Resistance Load

### 1.1 Original README.md from Lee Wiggins

This is all of the code, datasheets and design files for [my instructable](#)

- Arduino - It contains the Arduino code that we will be talking about here, within the dummy load folder. It also contains all of the 3rd party libraries I have used.
- Datasheets - It contains all of the datasheets for the major components used within the project.
- DesignSpark - I have used the opensource schematic and PCB design software for this project, its a fantastic free tool that has no limitations and I find it easier to 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.

### 1.2 Informations on this code from Daniel Caujolle-Bert

**Todo** write that!

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





## Chapter 2

# ATmega32U4 fuses settings

Unlike the Arduino™ 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
```

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

```
make fuses
```



## Chapter 3

# Hardware modifications for version 2.6

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

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



## Chapter 4

# User interface overview





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

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

You just need to rotate the encoder to define the desired value, accordingly to the focus : **Current**, **Power** or **Pulse Transient Time**.

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

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

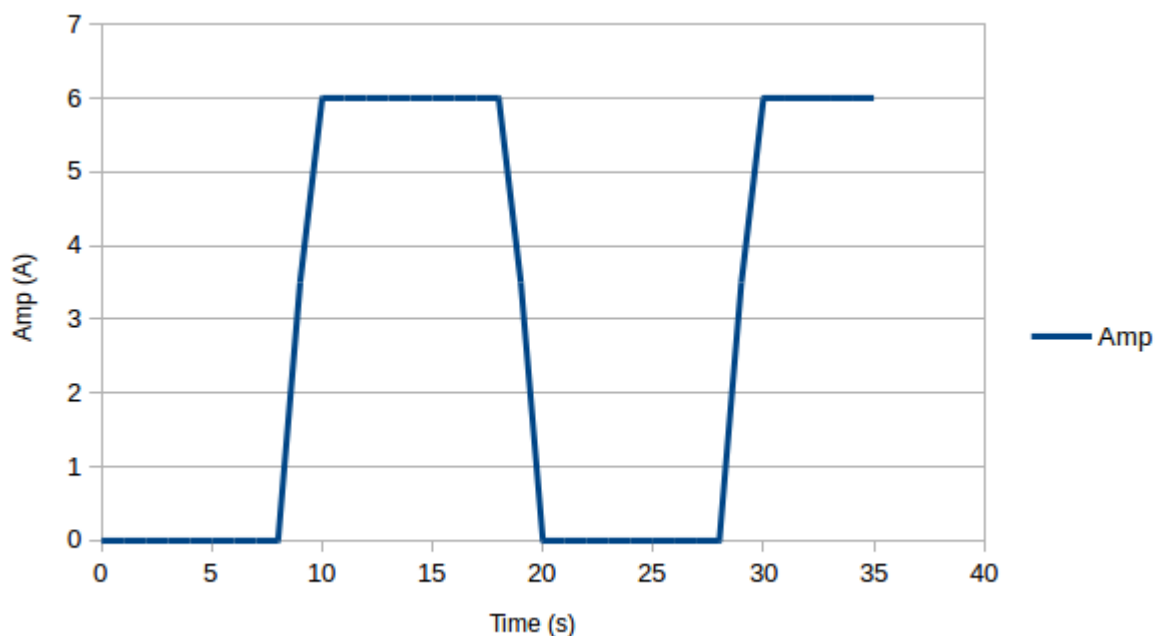
Multiplier	Glyph
x1	
x10	
x100	
x1000	

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

**Note**

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




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

**Note**

The maximum duration is **8.192s**

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

Feature	Glyph
Logging is running	
Encoder is locked	
USB remote controlled	

Over-Current 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 enableability 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 different kinds 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](#)

## 4.1 Extra control since v2.6

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

See [Hardware modifications for version 2.6](#)





## Chapter 5

# Remote Commands

See also [Logging data format](#).

### Note

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

### Warning

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

## 5.1 Get Identification

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

See [Return value](#).

## 5.2 Current setting getter

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

See [Return value](#).

## 5.3 Current setting setter

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

See [Return value](#).

## 5.4 Calibration values getter

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

See [Return value](#).

See [Calibration](#).

## 5.5 Calibration

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

See [Return value](#).

See [Calibration Process](#).

## 5.6 DAC value setter (calibration purpose)

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

### Note

This command has no effect outside calibration mode

See [Calibration](#).

See [Calibration Process](#).

## 5.7 Current readed getter

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

See [Return value](#).

## 5.8 Voltage readed getter

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

See [Return value](#).

## 5.9 Logging enability

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

See [Return value](#).

See [Logging data format](#).

## 5.10 Logging enability

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

Note

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

See [Return value](#).

See [Logging data format](#).

## 5.11 Pulse value getter

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

See [Return value](#).

## 5.12 Pulse value setter

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

See [Return value](#).

### 5.13 Input Relay status getter

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

See [Return value](#).

See [Hardware modifications for version 2.6](#).

### 5.14 Input Relay status setter

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

See [Return value](#).

See [Hardware modifications for version 2.6](#).

### 5.15 Return value

**:value:status:**

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

## Chapter 6

# Logging data format

See also [Remote Commands](#)

### Note

fields are comma separated

## 6.1 CSV logging format

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

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



## Chapter 7

# Calibration Process

- **Prerequisites:**

- **Hardware:**

- \* Amp-meter,
    - \* Volt-meter,
    - \* Power supply (**0..24V, 8A**)

- **Software:**

- \* A serial terminal emulator (e.g. “*HyperTerminal*” or “*Tera Term*” on Windows, “*minicom*” or “*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:

```
:CAL:ON  
  
:DAC:4095
```

Write down the **mA** value readed on the amp-meter to the “**mA<sub>max</sub>**” column. Now, type:

```
:DAC:0
```

Edit the [aDCLoad.h](#) source file, browse down the file, looking for the following line:

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

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

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

Calibration step 1 is now done.

- **Step 2: Voltage**

Select “*Step 2*” tab in the calibration spreadsheet file.

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

```
:CAL:ON
```

Set your power supply voltage output for each value in "**V<sub>set</sub>**" column, and write down the readed value in "**V<sub>read</sub>**" 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<sub>read</sub>**" 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 8

# Hierarchical Index

### 8.1 Class Hierarchy

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

aDCSettings::_eepromCalibrationValue_t . . . . .	27
aInputRelay . . . . .	56
aStepper . . . . .	61
aDCSettings . . . . .	36
aDCSettings::CalibrationData_t . . . . .	64
LiquidCrystal	
aLCD . . . . .	58
aDCDisplay . . . . .	27
aDCEngine . . . . .	31



## Chapter 9

# Class Index

### 9.1 Class List

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

<a href="#">aDCSettings::_eepromCalibrationValue_t</a>	Union to manipulate float/uint8_t [] calibration values . . . . .	27
<a href="#">aDCDisplay</a>	Class that handles LCD displaying . . . . .	27
<a href="#">aDCEngine</a>	Main class . . . . .	31
<a href="#">aDCSettings</a>	Class that handles settings . . . . .	36
<a href="#">aInputRelay</a>	Class that handles input relay . . . . .	56
<a href="#">aLCD</a>	LiquidDisplay extension class . . . . .	58
<a href="#">aStepper</a>	Class that handles increase/decrease step multiplier . . . . .	61
<a href="#">aDCSettings::CalibrationData_t</a>	Calibration values . . . . .	64



## Chapter 10

# File Index

### 10.1 File List

Here is a list of all files with brief descriptions:

aDCLoad.cpp	67
aDCLoad.h	72
CDC.cpp	84
HardwareSerial.cpp	84
HID.cpp	84
IPAddress.cpp	84
libraries.cpp	84
main.cpp	85
new.cpp	85
Print.cpp	85
sketch.cpp	85
Stream.cpp	85
Tone.cpp	86
USBCore.cpp	86
WInterrupts.c	86
wiring.c	86
wiring_analog.c	86
wiring_digital.c	86
wiring_pulse.c	86
wiring_shift.c	86
WMath.cpp	86
WString.cpp	86





## Class Documentation

[illegible]

- `~aDCDisplay ()`  
*Destructor.*
- `void setup ()`  
*Setup function. Should be called before any other member.*
- `void showBanner ()`  
*Display small banner.*
- `void updateField (aDCSettings::OperationMode_t, float, float, uint8_t, uint8_t)`  
*Update displayed field according to operation mode.*
- `void updateDisplay ()`  
*Update LCD display management function.*
- `void pingBacklight ()`  
*Reset backlight dimmer.*
- `bool isBacklightDimmed ()`  
*Check if backlight is dimmed.*

### Private Member Functions

- `void _dimBacklight (bool)`  
*Dim/undim backlight helper function.*
- `void _dimmingBacklight ()`  
*Backlight dimming.*
- `void _wakeupBacklight ()`  
*Backlight waker.*

### Private Attributes

- `aDCEngine * m_Parent`  
*Pointer to aDCEngine parent.*
- `bool m_dimmed`  
*Dimmed state storage.*
- `unsigned long m_dimmerTick`  
*Dimmer timeout tick counter.*
- `unsigned long m_nextUpdate`

#### 11.2.1 Detailed Description

Class that handles LCD displaying.

#### 11.2.2 Constructor & Destructor Documentation

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

Constructor.

Class to manage display output

## Parameters

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

## 11.2.2.2 aDCDisplay::~~aDCDisplay ( )

Destructor.

## 11.2.3 Member Function Documentation

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

Dim/undim backlight helper function.

## Parameters

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

## Returns

void

## 11.2.3.2 void aDCDisplay::\_dimmingBacklight ( ) [private]

Backlight dimming.

## Returns

void

## 11.2.3.3 void aDCDisplay::\_wakeupBacklight ( ) [private]

Backlight waker.

## Returns

void

## 11.2.3.4 bool aDCDisplay::isBacklightDimmed ( )

Check if backlight is dimmed.

## Returns

bool

#### 11.2.3.5 void aDCDisplay::pingBacklight ( )

Reset backlight dimmer.

##### Returns

void

#### 11.2.3.6 void aDCDisplay::setup ( )

Setup function. Should be called before any other member.

##### Returns

void

#### 11.2.3.7 void aDCDisplay::showBanner ( )

Display small banner.

##### Returns

void

#### 11.2.3.8 void aDCDisplay::updateDisplay ( )

Update LCD display management function.

Draw/Redraw on screen datas

##### Returns

void

#### 11.2.3.9 void aDCDisplay::updateField ( *aDCSettings::OperationMode\_t opMode*, *float vSet*, *float vRead*, *uint8\_t row*, *uint8\_t unit* )

Update displayed field according to operation mode.

##### Parameters

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

##### Returns

void

### 11.2.4 Member Data Documentation

#### 11.2.4.1 bool aDCDisplay::m\_dimmed [private]

Dimmed state storage.

11.2.4.2 unsigned long aDCDisplay::m\_dimmerTick [private]

Dimmer timeout tick counter.

11.2.4.3 unsigned long aDCDisplay::m\_nextUpdate [private]

11.2.4.4 aDCEngine\* aDCDisplay::m\_Parent [private]

Pointer to [aDCEngine](#) parent.

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

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

## 11.3 aDCEngine Class Reference

Main class.

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

Inherits [aDCDisplay](#).

### Public Member Functions

- [aDCEngine](#) (uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t=1)  
*Main engine constructor.*
- [~aDCEngine](#) ()  
*Destructor.*
- void [setup](#) ()  
*Setup function, should be called before any other member.*
- void [run](#) ()  
*Main loop function.*
- const [aDCSettings](#) \* [getSettings](#) () const  
*Return a pointer to settings instantiated class.*
- const ClickEncoder \* [getEncoder](#) () const
- void [setInput](#) (bool)

### Private Member Functions

- void [\\_handleButtonEvent](#) (ClickEncoder::Button)  
*Function that handles button clicking.*
- float [\\_getInputVoltage](#) ()  
*Function to read the input voltage and return a float number 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.*
- void [\\_handleLoggingAndRemote \(\)](#)  
*Check and handle remote control and data logging.*

### Private Attributes

- [aDCSettings m\\_Data](#)  
*Settings object.*
- ClickEncoder \* [m\\_encoder](#)  
*Encoder object.*
- uint8\_t [m\\_RXbuffer \[RXBUFFER\\_MAXLEN\]](#)  
*USB rx buffer.*
- uint8\_t [m\\_RXoffset](#)  
*USB rx buffer offset counter.*
- [aInputRelay m\\_inputRelay](#)

### Static Private Attributes

- static const uint8\_t [RXBUFFER\\_MAXLEN](#) = 64

### Friends

- class [aDCDisplay](#)

## 11.3.1 Detailed Description

Main class.

## 11.3.2 Constructor & Destructor Documentation

- 11.3.2.1 [aDCEngine::aDCEngine \( uint8\\_t rs, uint8\\_t enable, uint8\\_t d4, uint8\\_t d5, uint8\\_t d6, uint8\\_t d7, uint8\\_t cols, uint8\\_t rows, uint8\\_t irbtn, uint8\\_t ir, uint8\\_t enca, uint8\\_t encb, uint8\\_t encpb, uint8\\_t encsteps = 1 \)](#)

Main engine constructor.

Class to manage settings

Parameters

<i>rs</i>	uint8_t : <b>LCD RS pin</b>
<i>enable</i>	uint8_t : <b>LCD ENABLE pin</b>

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

### 11.3.2.2 aDCEngine::~~aDCEngine ( )

Destructor.

## 11.3.3 Member Function Documentation

### 11.3.3.1 void aDCEngine::\_adjustLoadCurrent ( ) [private]

Adjust current settings.

Returns

void

### 11.3.3.2 float aDCEngine::\_getADC ( uint8\_t *channel* ) [private]

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

Parameters

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

Returns

float : **readed value**

### 11.3.3.3 float aDCEngine::\_getInputVoltage ( ) [private]

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

Returns

float : **readed input voltage**

### 11.3.3.4 float aDCEngine::\_getMeasuredCurrent ( ) [private]

Function to measure the actual load current.

Returns

float : **readed current**

#### 11.3.3.5 `int16_t aDCEngine::_getTemp ( )` [private]

Function that read temperature from ADC channels.

Returns

`int16_t` : **temperature**

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

Function that handles button clicking.

Parameters

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

Returns

`void`

#### 11.3.3.7 `void aDCEngine::_handleLoggingAndRemote ( )` [private]

Check and handle remote control and data logging.

Returns

`void`

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

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

Parameters

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

Returns

`void`

#### 11.3.3.9 `void aDCEngine::_updateFanSpeed ( )` [private]

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

Returns

`void`

#### 11.3.3.10 `void aDCEngine::_updateLoadCurrent ( )` [private]

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

Returns

`void`



11.3.3.11 `const ClickEncoder * aDCEngine::getEncoder ( ) const`

11.3.3.12 `const aDCSettings * aDCEngine::getSettings ( ) const`

Return a pointer to settings instantiated class.

Returns

`const aDCSettings*`

11.3.3.13 `void aDCEngine::run ( )`

Main loop function.

Returns

`void`

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

Parameters

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

Returns

`void`

11.3.3.15 `void aDCEngine::setup ( )`

Setup function, should be called before any other member.

Returns

`void`

## 11.3.4 Friends And Related Function Documentation

11.3.4.1 `friend class aDCDisplay [friend]`

## 11.3.5 Member Data Documentation

11.3.5.1 `aDCSettings aDCEngine::m_Data [private]`

Settings object.

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

Encoder object.

11.3.5.3 `alInputRelay aDCEngine::m_inputRelay [private]`

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

USB rx buffer.

11.3.5.5 `uint8_t aDCEngine::m_RXOffset` [private]

USB rx buffer offset counter.

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

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

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

## 11.4 aDCSettings Class Reference

Class that handles settings.

`#include <aDCLoad.h>`

Inherits [aStepper](#).

### Classes

- union [\\_eepromCalibrationValue\\_t](#)  
*Union to manipulate float/uint8\_t [] calibration values.*
- struct [CalibrationData\\_t](#)  
*Calibration values.*

### Public Types

- enum [OperationMode\\_t](#) { OPERATION\_MODE\_READ, OPERATION\_MODE\_SET, OPERATION\_MODE\_UNKNOWN }  
*Operation mode enumeration.*
- enum [SelectionMode\\_t](#) { SELECTION\_MODE\_CURRENT, SELECTION\_MODE\_POWER, SELECTION\_MODE\_PULSE, SELECTION\_MODE\_UNKNOWN }  
*Selection settings mode enumeration.*
- enum [DisplayMode\\_t](#) { DISPLAY\_MODE\_VALUES, DISPLAY\_MODE\_SETUP, DISPLAY\_MODE\_UNKNOWN }  
*Display mode enumeration.*
- enum [SettingError\\_t](#) { SETTING\_ERROR\_OVERSIZED, SETTING\_ERROR\_UNDERSIZED, SETTING\_ERROR\_VALID }  
*Setting error enumeration.*
- enum [CalibrationValues\\_t](#) { CALIBRATION\_VOLTAGE, CALIBRATION\_READ\_CURRENT, CALIBRATION\_DAC\_CURRENT, CALIBRATION\_VOLTAGE\_DROP, CALIBRATION\_MAX }  
*Calibration offset enumeration.*

### Public Member Functions

- [aDCSettings\(aDCEngine \\*\)](#)  
*Constructor.*
- [~aDCSettings\(\)](#)  
*Destructor.*

- [SettingError\\_t setVoltage](#) (float)  
*Voltage setter.*
- float [getVoltage](#) ()  
*Voltage getter.*
- [SettingError\\_t setCurrent](#) (float, [OperationMode\\_t](#))  
*Current setter.*
- float [getCurrent](#) ([OperationMode\\_t](#))  
*Current getter.*
- [SettingError\\_t setPulse](#) (float)  
*Pulse setter.*
- float [getPulse](#) ()  
*Pulse getter.*
- bool [isPulseEnabled](#) ()  
*Returns pulse enableity.*
- void [enablePulse](#) (bool)  
*Set pulse enableity.*
- bool [isPulseHigh](#) ()  
*Returns if pulse is high (load should operate)*
- void [setPulseHigh](#) (bool)  
*Set pulse high/low state value.*
- [SettingError\\_t setPower](#) (float, [OperationMode\\_t](#))  
*Power setter.*
- float [getPower](#) ([OperationMode\\_t](#))  
*Power getter.*
- void [updateValuesFromMode](#) (float, [SelectionMode\\_t](#))  
*Update values setting (Current, Resistance, Power) according to selection mode. Sanity checking is also performed.*
- void [setTemperature](#) (uint16\_t)  
*Temperature readed setter.*
- uint16\_t [getTemperature](#) ()  
*Temperature readed getter.*
- void [setFanSpeed](#) (uint16\_t)  
*Fan speed setter.*
- uint16\_t [getFanSpeed](#) ()  
*Fan speed getter.*
- void [setCurrentDAC](#) (uint16\_t)  
*Current DAC value setter.*
- uint16\_t [getCurrentDAC](#) ()  
*Current DAC value getter.*
- void [setSelectionMode](#) ([SelectionMode\\_t](#), bool=false)  
*Selection mode setter.*
- [SelectionMode\\_t](#) [getSelectionMode](#) ()  
*Selection mode getter.*
- [SelectionMode\\_t](#) [getPrevNextMode](#) ([SelectionMode\\_t](#)=[SELECTION\\_MODE\\_UNKNOWN](#), bool=true)  
*Get the next selection mode, according to "origin", if any provided.*
- void [setDisplayMode](#) ([DisplayMode\\_t](#))  
*Display mode setter.*
- [DisplayMode\\_t](#) [getDisplayMode](#) ()  
*Display mode getter.*
- void [setEncoderPosition](#) (int32\_t)  
*Encoder position setter.*
- void [incEncoderPosition](#) (int32\_t=1)

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

  - `int32_t getEncoderPosition ()`  
*Encoder position getter.*
  - `void setOperationMode (OperationMode_t)`  
*Operation mode setter.*
  - `OperationMode_t getOperationMode ()`  
*Operation mode getter.*
  - `void updateOperationMode ()`  
*Automatic timeouted toggle between OPERATION\_SET and OPERATION\_READ.*
  - `void pingOperationMode ()`  
*Reset timeout while in OPERATION\_SET mode.*
  - `void pingAutolock ()`  
*Reset autolock timeout.*
  - `bool isAutolocked ()`  
*Check if autolock is enabled and performs.*
  - `void 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_PULSE_SETS = 1 << 3`
- `static const uint16_t DATA_POWER_SETS = 1 << 5`  
*bit-field storage: Power sets*
- `static const uint16_t DATA_POWER_READ = 1 << 6`

- bit-field storage: Power readed*
- static const uint16\_t [DATA\\_TEMPERATURE](#) = 1 << 7
- bit-field storage: Temperature readed*
- static const uint16\_t [DATA\\_SELECTION](#) = 1 << 8
- bit-field storage: Selection mode sets*
- static const uint16\_t [DATA\\_DISPLAY](#) = 1 << 9
- bit-field storage: Display mode sets*
- static const uint16\_t [DATA\\_ENCODER](#) = 1 << 10
- bit-field storage: Encoder position sets*
- static const uint16\_t [DATA\\_OPERATION](#) = 1 << 11
- bit-field storage: Operation mode sets*
- static const uint16\_t [DATA\\_IN\\_CALIBRATION](#) = 1 << 12
- bit-field storage: In calibration mode*

### Private Member Functions

- [SettingError\\_t \\_setValue](#) ([OperationMode\\_t](#), uint16\_t, float, float &, float &, float)
- Value setter.*
- uint8\_t [\\_crc8](#) (const uint8\_t \*, uint8\_t)
- CRC8 computation.*
- void [\\_eepromCalibrationRestore](#) (int16\_t, [CalibrationData\\_t](#) &)
- Read data from EEPROM, used to restore calibration data.*
- void [\\_eepromCalibrationBackup](#) (int16\_t, [CalibrationData\\_t](#))
- Save data to EEPROM, used to restore calibration data.*
- bool [\\_eepromCheckForMagicNumbers](#) ()
- Check for EEPROM magic numbers.*
- void [\\_eepromWriteMagicNumbers](#) ()
- Write magic numbers into EEPROM.*
- void [\\_eepromReset](#) ()
- Reset all stored parameters into EEPROM.*
- void [\\_eepromRestore](#) ()
- Restore value from EEPROM.*
- void [\\_enableData](#) (uint16\_t, bool)
- Enable a bit in the m\_datas bit-field storage.*
- void [\\_enableDataCheck](#) (uint16\_t, bool)
- Enable a bit in the m\_datas bit-field storage, checking for previous state.*

### Private Attributes

- [aDCEngine](#) \* [m\\_Parent](#)
- float [m\\_readVoltage](#)
- voltage storage*
- float [m\\_setsCurrent](#)
- float [m\\_readCurrent](#)
- current storage*
- float [m\\_setsPulse](#)
- Pulse time length (ms)*
- float [m\\_setsPower](#)
- float [m\\_readPower](#)
- power storage*

- [uint16\\_t m\\_readTemperature](#)  
*temperature storage*
- [uint16\\_t m\\_fanSpeed](#)  
*fan speed storage*
- [uint16\\_t m\\_currentDAC](#)  
*current DAC value*
- [OperationMode\\_t m\\_operationMode](#)  
*operation mode*
- [SelectionMode\\_t m\\_mode](#)  
*selection mode*
- [int32\\_t m\\_encoderPos](#)  
*encoder position (yeah, 32bits, due to resistance max val (12000 \* 1000);*
- [DisplayMode\\_t m\\_dispMode](#)  
*display mode*
- unsigned long [m\\_lockTick](#)  
*tick count storage, for autolock feature*
- unsigned long [m\\_operationTick](#)  
*tick count storage, for SET/READ operation*
- [uint16\\_t m\\_features](#)  
*boolean features storage*
- [uint16\\_t m\\_datas](#)  
*boolean displays data storage*
- [CalibrationData\\_t m\\_calibrationValues \[CALIBRATION\\_MAX\]](#)  
*Calibration datas, restored from EEPROM.*
- volatile bool [m\\_pulseEnabled](#)  
*Pulse is enabled.*
- volatile bool [m\\_pulseHigh](#)  
*Pulse is high ?*

### 11.4.1 Detailed Description

Class that handles settings.

### 11.4.2 Member Enumeration Documentation

#### 11.4.2.1 enum `aDCSettings::CalibrationValues_t`

Calibration offset enumeration.

Used to get/set calibration values

Enumerator

**`CALIBRATION_VOLTAGE`** Voltage calibration offset value.

**`CALIBRATION_READ_CURRENT`** Readed Current calibration offset value.

**`CALIBRATION_DAC_CURRENT`** DAC Current calibration offset value.

**`CALIBRATION_VOLTAGE_DROP`** Voltage drop calibration offset value.

**`CALIBRATION_MAX`** Maximum offset in calibration array.

## 11.4.2.2 enum aDCSettings::DisplayMode\_t

Display mode enumeration.

Enumerator

**DISPLAY\_MODE\_VALUES** Display read/set values mode.  
**DISPLAY\_MODE\_SETUP** Display settings mode.  
**DISPLAY\_MODE\_UNKNOWN** Display in undefined (internal)

## 11.4.2.3 enum aDCSettings::OperationMode\_t

Operation mode enumeration.

Enumerator

**OPERATION\_MODE\_READ** Reading values.  
**OPERATION\_MODE\_SET** Settings values.  
**OPERATION\_MODE\_UNKNOWN** Unset (internal)

## 11.4.2.4 enum aDCSettings::SelectionMode\_t

Selection settings mode enumeration.

Enumerator

**SELECTION\_MODE\_CURRENT** Current selected.  
**SELECTION\_MODE\_POWER** Power selected.  
**SELECTION\_MODE\_PULSE** Pulse selected.  
**SELECTION\_MODE\_UNKNOWN** Nothing selected (internal)

## 11.4.2.5 enum aDCSettings::SettingError\_t

Setting error enumeration.

Enumerator

**SETTING\_ERROR\_OVERSIZED** Setting value is oversized.  
**SETTING\_ERROR\_UNDERSIZED** Setting value is undersized.  
**SETTING\_ERROR\_VALID** Setting value is valid.

## 11.4.3 Constructor &amp; Destructor Documentation

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

Constructor.

## Parameters

<i>parent</i>	aDCEngine* : <b>Pointer to aDCEngine parent</b>
---------------	---

## 11.4.3.2 aDCSettings::~~aDCSettings ( )

Destructor.

## 11.4.4 Member Function Documentation

11.4.4.1 uint8\_t aDCSettings::\_crc8 ( const uint8\_t\* *addr*, uint8\_t *len* ) [private]

CRC8 computation.

Code took from [http://www.pjrc.com/teensy/td\\_libs\\_OneWire.html](http://www.pjrc.com/teensy/td_libs_OneWire.html)

## Parameters

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

## Returns

uint8\_t : **CRC**

11.4.4.2 void aDCSettings::\_eepromCalibrationBackup ( int16\_t *addr*, CalibrationData\_t *cal* ) [private]

Save data to EEPROM, used to restore calibration data.

## Parameters

<i>addr</i>	int16_t : <b>start address location</b>
<i>cal</i>	<a href="#">CalibrationData_t</a> & : <b>destination</b>

## Returns

void

11.4.4.3 void aDCSettings::\_eepromCalibrationRestore ( int16\_t *addr*, CalibrationData\_t & *cal* ) [private]

Read data from EEPROM, used to restore calibration data.

## Parameters

<i>addr</i>	int16_t : <b>start address location</b>
<i>cal</i>	<a href="#">CalibrationData_t</a> & : <b>destination</b>

## Returns

void

## 11.4.4.4 bool aDCSettings::\_eepromCheckForMagicNumbers ( ) [private]

Check for EEPROM magic numbers.



**Returns**

bool

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

**11.4.4.5 void aDCSettings::\_eepromReset ( ) [private]**

Reset all stored parameters into EEPROM.

**Returns**

void

**11.4.4.6 void aDCSettings::\_eepromRestore ( ) [private]**

Restore value from EEPROM.

**Returns**

void

**11.4.4.7 void aDCSettings::\_eepromWriteMagicNumbers ( ) [private]**

Write magic numbers into EEPROM.

**Returns**

void

**11.4.4.8 void aDCSettings::\_enableData ( uint16\_t *bit*, bool *enable* ) [private]**

Enable a bit in the m\_datas bit-field storage.

**Parameters**

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

**Returns**

void

**11.4.4.9 void aDCSettings::\_enableDataCheck ( uint16\_t *bit*, bool *enable* ) [private]**

Enable a bit in the m\_datas bit-field storage, checking for previous state.

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

**Parameters**

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

**Returns**

void

11.4.4.10 **aDCSettings::SettingError\_t aDCSettings::setValue ( OperationMode\_t mode, uint16\_t bit, float value, float & sets, float & read, float maximum ) [private]**

Value setter.

**Parameters**

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

**Returns**SettingError\_t : **validity result**

11.4.4.11 **void aDCSettings::backupCalibration ( )**

Backup calibration data into EEPROM.

**Returns**

void

11.4.4.12 **void aDCSettings::enableAlarm ( uint16\_t aBit )**

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

**Parameters**

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

**Returns**

void

11.4.4.13 **void aDCSettings::enableFeature ( uint16\_t feature, bool enable = true )**

Helper function to manage bit-field features.

**Parameters**

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

## Returns

void

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

Set pulse enableity.

## Parameters

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

## Returns

void

## 11.4.4.15 bool aDCSettings::getCalibrationMode ( )

Calibration mode enableity getter.

## Returns

bool : **enableity**11.4.4.16 void aDCSettings::getCalibrationValues ( CalibrationValues\_t *calsection*, CalibrationData\_t & *data* )Calibration data getter, according to *calsection* argument.

## Parameters

<i>calsection</i>	CalibrationValues_t : <b>calibration parameter</b>
<i>data</i>	<a href="#">CalibrationData_t</a> & : <b>calibration data</b>

## Returns

void

11.4.4.17 float aDCSettings::getCurrent ( OperationMode\_t *mode* )

Current getter.

## Parameters

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

## Returns

float : **current**

#### 11.4.4.18 uint16\_t aDCSettings::getCurrentDAC ( )

Current DAC value getter.

Returns

uint16\_t : **current DAC value**

#### 11.4.4.19 aDCSettings::DisplayMode\_t aDCSettings::getDisplayMode ( )

Display mode getter.

Returns

DisplayMode\_t : **Display mode**

#### 11.4.4.20 int32\_t aDCSettings::getEncoderPosition ( )

Encoder position getter.

Returns

int32\_t : **encoder position**

#### 11.4.4.21 uint16\_t aDCSettings::getFanSpeed ( )

Fan speed getter.

Returns

uint16\_t : **DAC speed value**

#### 11.4.4.22 aDCSettings::OperationMode\_t aDCSettings::getOperationMode ( )

Operation mode getter.

Returns

OperationMode\_t : **operation mode**

#### 11.4.4.23 float aDCSettings::getPower ( OperationMode\_t mode )

Power getter.

Parameters

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

Returns

float : **power**

#### 11.4.4.24 aDCSettings::SelectionMode\_t aDCSettings::getPrevNextMode ( aDCSettings::SelectionMode\_t origin = SELECTION\_MODE\_UNKNOWN, bool next = true )

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

## Parameters

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

## Returns

SelectionMode\_t : **next selection mode**

11.4.4.25 float aDCSettings::getPulse ( )

Pulse getter.

## Returns

float : **pulse (ms)**

11.4.4.26 aDCSettings::SelectionMode\_t aDCSettings::getSelectionMode ( )

Selection mode getter.

## Returns

SelectionMode\_t : **current selection mode**

11.4.4.27 uint16\_t aDCSettings::getTemperature ( )

Temperature readed getter.

## Returns

uint16\_t : **temperature**

11.4.4.28 float aDCSettings::getVoltage ( )

Voltage getter.

## Returns

float : **voltage**

11.4.4.29 void aDCSettings::incEncoderPosition ( int32\_t v = 1 )

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

## Parameters

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

## Returns

void

#### 11.4.4.30 bool aDCSettings::isAutolocked ( )

Check if autolock is enabled and performs.

##### Returns

bool

#### 11.4.4.31 bool aDCSettings::isDataEnabled ( uint16\_t *bit* )

Get bit enability in m\_datas bit-field storage.

##### Parameters

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

##### Returns

bool

#### 11.4.4.32 bool aDCSettings::isFeatureEnabled ( uint16\_t *feature* )

Check if FEATURE is enabled.

##### Parameters

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

##### Returns

bool

#### 11.4.4.33 bool aDCSettings::isPulseEnabled ( )

Returns pulse enability.

##### Returns

bool

#### 11.4.4.34 bool aDCSettings::isPulseHigh ( )

Returns if pulse is high (load should operate)

##### Returns

bool

#### 11.4.4.35 void aDCSettings::pingAutolock ( )

Reset autolock timeout.

##### Returns

void

## 11.4.4.36 void aDCSettings::pingOperationMode ( )

Reset timeout while in OPERATION\_SET mode.

Returns

void

## 11.4.4.37 void aDCSettings::restoreCalibration ( )

Restore calibration data from EEPROM.

Returns

void

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

Calibration mode enable setter.

Parameters

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

Returns

void

11.4.4.39 void aDCSettings::setCalibrationValues ( CalibrationValues\_t *calsection*, CalibrationData\_t *data* )

Calibration data setter, according to *calsection* argument.

Parameters

<i>calsection</i>	CalibrationValues_t : <b>calibration parameter</b>
<i>data</i>	<a href="#">CalibrationData_t</a> & : <b>calibration data</b>

Returns

void

11.4.4.40 aDCSettings::SettingError\_t aDCSettings::setCurrent ( float *v*, OperationMode\_t *mode* )

Current setter.

Parameters

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

Returns

[aDCSettings::SettingError\\_t](#)

11.4.4.41 void aDCSettings::setCurrentDAC ( uint16\_t *dac* )

Current DAC value setter.

## Parameters

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

## Returns

void

**11.4.4.42 void aDCSettings::setDisplayMode ( DisplayMode\_t d )**

Display mode setter.

## Parameters

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

## Returns

void

**11.4.4.43 void aDCSettings::setEncoderPosition ( int32\_t v )**

Encoder position setter.

## Parameters

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

## Returns

void

**11.4.4.44 void aDCSettings::setFanSpeed ( uint16\_t v )**

Fan speed setter.

## Parameters

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

## Returns

void

**11.4.4.45 void aDCSettings::setOperationMode ( OperationMode\_t m )**

Operation mode setter.

## Parameters

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

## Returns

void



11.4.4.46 aDCSettings::SettingError\_t aDCSettings::setPower ( float *v*, OperationMode\_t *mode* )

Power setter.

## Parameters

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

## Returns

[aDCSettings::SettingError\\_t](#)

#### 11.4.4.47 aDCSettings::SettingError\_t aDCSettings::setPulse ( float *v* )

Pulse setter.

## Parameters

<i>v</i>	float : <b>pulse (ms)</b>
----------	---------------------------

## Returns

[aDCSettings::SettingError\\_t](#)

< Sleep well honey

< Wake up son!

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

Set pulse high/low state value.

## Parameters

<i>high</i>	bool : <b>pulse state</b>
-------------	---------------------------

## Returns

void

#### 11.4.4.49 void aDCSettings::setSelectionMode ( SelectionMode\_t *m*, bool *force* = false )

Selection mode setter.

## Parameters

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

## Returns

void

#### 11.4.4.50 void aDCSettings::setTemperature ( uint16\_t *v* )

Temperature readed setter.

## Parameters

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

## Returns

void

11.4.4.51 aDCSettings::SettingError\_t aDCSettings::setVoltage ( float *v* )

Voltage setter.

## Parameters

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

## Returns

[aDCSettings::SettingError\\_t](#)11.4.4.52 void aDCSettings::syncData ( uint16\_t *bit* )

Clear a bit in m\_datas bit-field storage.

## Parameters

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

## Returns

void

## 11.4.4.53 void aDCSettings::updateOperationMode ( )

Automatic timeouted toggle between OPERATION\_SET and OPERATION\_READ.

## Returns

void

11.4.4.54 void aDCSettings::updateValuesFromMode ( float *v*, SelectionMode\_t *mode* )

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

## Parameters

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

## Returns

void

### 11.4.5 Member Data Documentation

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

bit-field storage: Current readed

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

bit-field storage: Current sets

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

bit-field storage: Display mode sets

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

bit-field storage: Encoder position sets

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

bit-field storage: In calibration mode

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

bit-field storage: Operation mode sets

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

bit-field storage: Power readed

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

bit-field storage: Power sets

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

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

bit-field storage: Selection mode sets

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

bit-field storage: Temperature readed

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

bit-field storage: Voltage readed

11.4.5.13 **CalibrationData\_t** aDCSettings::m\_calibrationValues[**CALIBRATION\_MAX**] [private]

Calibration datas, restored from EEPROM.

11.4.5.14 **uint16\_t** aDCSettings::m\_currentDAC [private]

current DAC value

11.4.5.15 **uint16\_t** aDCSettings::m\_datas [private]

boolean displays data storage

11.4.5.16 **DisplayMode\_t** aDCSettings::m\_dispMode [private]

display mode

See Also

DisplayMode

11.4.5.17 **int32\_t** aDCSettings::m\_encoderPos [private]

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

11.4.5.18 **uint16\_t** aDCSettings::m\_fanSpeed [private]

fan speed storage

11.4.5.19 **uint16\_t** aDCSettings::m\_features [private]

boolean features storage

11.4.5.20 **unsigned long** aDCSettings::m\_lockTick [private]

tick count storage, for autolock feature

11.4.5.21 **SelectionMode\_t** aDCSettings::m\_mode [private]

selection mode

See Also

SelectionMode

11.4.5.22 **OperationMode\_t** aDCSettings::m\_operationMode [private]

operation mode

See Also

OperationMode

11.4.5.23 unsigned long aDCSettings::m\_operationTick [private]

tick count storage, for SET/READ operation

11.4.5.24 aDCEngine\* aDCSettings::m\_Parent [private]

11.4.5.25 volatile bool aDCSettings::m\_pulseEnabled [private]

Pulse is enabled.

11.4.5.26 volatile bool aDCSettings::m\_pulseHigh [private]

Pulse is high ?

11.4.5.27 float aDCSettings::m\_readCurrent [private]

current storage

11.4.5.28 float aDCSettings::m\_readPower [private]

power storage

11.4.5.29 uint16\_t aDCSettings::m\_readTemperature [private]

temperature storage

11.4.5.30 float aDCSettings::m\_readVoltage [private]

voltage storage

11.4.5.31 float aDCSettings::m\_setsCurrent [private]

11.4.5.32 float aDCSettings::m\_setsPower [private]

11.4.5.33 float aDCSettings::m\_setsPulse [private]

Pulse time length (ms)

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

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

## 11.5 aInputRelay Class Reference

Class that handles input relay.

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

## Public Member Functions

- [aInputRelay](#) (uint8\_t, uint8\_t)  
*Constructor.*
- [~aInputRelay](#) ()  
*Destructor.*
- void [service](#) ()  
*Function to check button press checking, should be called inside the main loop.*
- bool [isInput](#) ()  
*Returns the relay's state.*
- void [setInput](#) (bool=true)  
*Set relay's input state.*
- bool [isClicked](#) ()  
*Returns true if button has been clicked (pressed then released)*

## Private Attributes

- uint8\_t [m\\_btnPin](#)
- uint8\_t [m\\_relayPin](#)
- uint8\_t [m\\_btnState](#)
- bool [m\\_clicked](#)
- bool [m\\_isON](#)

### 11.5.1 Detailed Description

Class that handles input relay.

### 11.5.2 Constructor & Destructor Documentation

#### 11.5.2.1 aInputRelay::aInputRelay ( uint8\_t btnPin, uint8\_t relayPin )

Constructor.

Class to manage input relay

Parameters

<i>btnPin</i>	uint8_t : <b>Button pin</b>
<i>relayPin</i>	uint8_t : <b>Relay pin</b>

#### 11.5.2.2 aInputRelay::~~aInputRelay ( )

Destructor.

### 11.5.3 Member Function Documentation

#### 11.5.3.1 bool aInputRelay::isClicked ( )

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

Returns

bool

### 11.5.3.2 `bool alnputRelay::isInput ( )`

Returns the relay's state.

Returns

`bool`

### 11.5.3.3 `void alnputRelay::service ( )`

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

Returns

`void`

### 11.5.3.4 `void alnputRelay::setInput ( bool on =true )`

Set relay's input state.

Parameters

<i>on</i>	<code>bool</code>
-----------	-------------------

Returns

`void`

## 11.5.4 Member Data Documentation

11.5.4.1 `uint8_t alnputRelay::m_btnPin` [private]

11.5.4.2 `uint8_t alnputRelay::m_btnState` [private]

11.5.4.3 `bool alnputRelay::m_clicked` [private]

11.5.4.4 `bool alnputRelay::m_isON` [private]

11.5.4.5 `uint8_t alnputRelay::m_relayPin` [private]

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

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

## 11.6 aLCD Class Reference

LiquidDisplay extension class.

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

Inherits LiquidCrystal.

Inherited by [aDCDisplay](#).



## Public Member Functions

- **aLCD** (uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t, uint8\_t)  
*Constructor.*
- **~aLCD** ()  
*Destructor.*
- void **begin** (uint8\_t, uint8\_t)  
*Initialize LCD screen size.*
- void **setCursor** (uint8\_t, uint8\_t)  
*Set cursor position.*
- void **printCenter** (const char \*)  
*Print centered string to current row.*
- void **printCenter** (const \_\_FlashStringHelper \*)  
*Print centered string to current row.*
- void **clearValue** (uint8\_t, int=0)  
*Clear displayed value, from given row, stopping at value end field - destMinus.*

## Private Attributes

- uint8\_t **m\_cols**
- uint8\_t **m\_rows**  
*LCD sizes.*
- uint8\_t **m\_curCol**
- uint8\_t **m\_curRow**  
*Current cursor position.*

### 11.6.1 Detailed Description

LiquidDisplay extension class.

### 11.6.2 Constructor & Destructor Documentation

11.6.2.1 **aLCD::aLCD** ( uint8\_t *rs*, uint8\_t *enable*, uint8\_t *d4*, uint8\_t *d5*, uint8\_t *d6*, uint8\_t *d7*, uint8\_t *cols*, uint8\_t *rows* )

Constructor.

LiquidDisplay extension

Parameters

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

11.6.2.2 **aLCD::~~aLCD** ( )

Destructor.

### 11.6.3 Member Function Documentation

#### 11.6.3.1 void aLCD::begin ( uint8\_t cols, uint8\_t lines )

Initialize LCD screen size.

Parameters

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

Returns

void

#### 11.6.3.2 void aLCD::clearValue ( uint8\_t row, int destMinus = 0 )

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

Parameters

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

Returns

void

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

Print centered string to current row.

Parameters

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

Returns

void

#### 11.6.3.4 void aLCD::printCenter ( const \_\_FlashStringHelper \* ifsh )

Print centered string to current row.

Parameters

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

Returns

void

#### 11.6.3.5 void aLCD::setCursor ( uint8\_t col, uint8\_t row )

Set cursor positon.

## Parameters

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

## Returns

void

## 11.6.4 Member Data Documentation

11.6.4.1 uint8\_t aLCD::m\_cols [private]

11.6.4.2 uint8\_t aLCD::m\_curCol [private]

11.6.4.3 uint8\_t aLCD::m\_curRow [private]

Current cursor position.

11.6.4.4 uint8\_t aLCD::m\_rows [private]

LCD sizes.

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

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

## 11.7 aStepper Class Reference

Class that handles increase/decrease step multiplier.

#include &lt;aDCLoad.h&gt;

Inherited by [aDCSettings](#).

## Public Member Functions

- [aStepper](#) ()  
*Constructor.*
- [~aStepper](#) ()  
*Destructor.*
- void [increment](#) ()  
*Increment value, check for boundaries.*
- uint8\_t [getValue](#) ()  
*Value getter.*
- void [reset](#) ()  
*Reset value.*
- int16\_t [getMult](#) ()  
*Value getter, according to multiple.*
- int16\_t [getValueFromMode](#) (uint8\_t)  
*Get value according to selection mode.*
- bool [isSynced](#) ()

*Synchronize value.*

- void `sync` ()

*Check if value is synchronized.*

## Private Member Functions

- int16\_t `_pow` (int, int)

*Small implementation of pow() math function.*

## Private Attributes

- uint8\_t `m_inc`
- uint8\_t `m_incPrev`

*Stepper counters.*

## Static Private Attributes

- static const uint8\_t `MAX_VALUE` = 3

### 11.7.1 Detailed Description

Class that handles increase/decrease step multiplier.

Classes declarations

### 11.7.2 Constructor & Destructor Documentation

#### 11.7.2.1 aStepper::aStepper ( )

Constructor.

#### 11.7.2.2 aStepper::~~aStepper ( )

Destructor.

### 11.7.3 Member Function Documentation

#### 11.7.3.1 int16\_t aStepper::\_pow ( int *base*, int *exp* ) [inline], [private]

Small implementation of pow() math function.

Parameters

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

Returns

int16\_t : **result**

### 11.7.3.2 int16\_t aStepper::getMult ( )

Value getter, according to multiple.

Returns

int16\_t

### 11.7.3.3 uint8\_t aStepper::getValue ( )

Value getter.

Returns

uint8\_t

### 11.7.3.4 int16\_t aStepper::getValueFromMode ( uint8\_t mode )

Get value according to selection mode.

Parameters

<i>mode</i>	uint16_t : <b>Selection mode (will be typecasted to <a href="#">aDCSettings::SelectionMode_t</a>)</b>
-------------	---

Returns

int16\_t

### 11.7.3.5 void aStepper::increment ( )

Increment value, check for boundaries.

Returns

void

### 11.7.3.6 bool aStepper::isSynced ( )

Synchronize value.

Returns

bool

### 11.7.3.7 void aStepper::reset ( )

Reset value.

Returns

void

### 11.7.3.8 void aStepper::sync ( )

Check if value is synchronized.

#### Returns

void

## 11.7.4 Member Data Documentation

### 11.7.4.1 uint8\_t aStepper::m\_inc [private]

### 11.7.4.2 uint8\_t aStepper::m\_incPrev [private]

Stepper counters.

### 11.7.4.3 const uint8\_t aStepper::MAX\_VALUE = 3 [static], [private]

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

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

## 11.8 aDCSettings::CalibrationData\_t Struct Reference

Calibration values.

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

### Public Attributes

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

### 11.8.1 Detailed Description

Calibration values.

Contains Slope and Offset float values

## 11.8.2 Member Data Documentation

### 11.8.2.1 float aDCSettings::CalibrationData\_t::offset

Offset value.

## 11.8.2.2 float aDCSettings::CalibrationData\_t::slope

Slope value.

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

- [aDCLoad.h](#)





## Chapter 12

# File Documentation

### 12.1 aDCLoad.cpp File Reference

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

#### Functions

- void [serialPrint](#) (unsigned long n, int base)  
*Serial printing.*
- void [serialWrite](#) (char c)  
*Serial printing.*
- void [serialPrint](#) (const char str[])  
*Serial printing.*
- void [serialPrint](#) (char c)  
*Serial printing.*
- void [serialPrint](#) (int n, int16\_t base=DEC)  
*Serial printing.*
- void [serialFlush](#) ()  
*Flush serial buffer (TX)*
- void [serialPrintln](#) ()  
*Serial printing.*
- void [serialPrint](#) (double n, int digits)  
*Serial printing.*
- void [serialPrintln](#) (double n, int digits)  
*Serial printing.*
- void [serialPrintln](#) (char c)  
*Serial printing.*
- int16\_t [serialAvailable](#) ()  
*Get the number of bytes (characters) available for reading from the serial port.*
- uint8\_t [serialRead](#) ()

- Reads incoming serial data.*
- `int8_t getNumericalLength` (float n, uint8\_t len=3)  
*Get float string length, according to decimal man length.*
- `float floatRounding` (float f)  
*Float number rounding, extensively used.*
- `static void timer1ISR` (void)  
*ISR callback function for Timer1, used for encoder.*
- `static void timer3ISR` (void)  
*ISR callback function for Timer3, used for pulse feature.*

## Variables

- `ADCEngine * pThis` = NULL

### 12.1.1 Detailed Description

#### Copyright

Copyright (C) 2014-2015 F1RMB, Daniel Caujolle-Bert [f1rmb.daniel@gmail.com](mailto:f1rmb.daniel@gmail.com)

Copyright (C) 2014 Lee Wiggins [lee@wigweb.com.au](mailto: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](mailto:f1rmb.daniel@gmail.com)

Lee Wiggins [lee@wigweb.com.au](mailto:lee@wigweb.com.au)

### 12.1.2 Function Documentation

#### 12.1.2.1 float floatRounding ( float f )

Float number rounding, extensively used.

Float rounding functionWe just get rid of +4 decimal values

## Parameters

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

## Returns

float : **rounded value**

12.1.2.2 int8\_t getNumericalLength ( float *n*, uint8\_t *len* = 3 )

Get float string length, according to decimal man length.

Our float to string format function

## Parameters

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

## Returns

int8\_t : **numeric value length**

## 12.1.2.3 int16\_t serialAvailable ( )

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

## Returns

int16\_t

## 12.1.2.4 void serialFlush ( )

Flush serial buffer (TX)

## Returns

void

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

Serial printing.

Implement our serial print function to save ~300ko

## Parameters

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

## Returns

void

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

Serial printing.

## Parameters

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

## Returns

void

12.1.2.7 void serialPrint ( char *c* )

Serial printing.

## Parameters

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

## Returns

void

12.1.2.8 void serialPrint ( int *n*, int16\_t *base* = DEC )

Serial printing.

## Parameters

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

## Returns

void

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

Serial printing.

## Parameters

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

## Returns

void

## 12.1.2.10 void serialPrintln ( )

Serial printing.

## Returns

void

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

Serial printing.

## Parameters

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

## Returns

void

12.1.2.12 void serialPrintln ( char *c* )

Serial printing.

## Parameters

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

## Returns

void

## 12.1.2.13 uint8\_t serialRead ( )

Reads incoming serial data.

## Returns

uint8\_t

12.1.2.14 void serialWrite ( char *c* )

Serial printing.

## Parameters

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

## Returns

void

## 12.1.2.15 static void timer1ISR ( void ) [static]

ISR callback function for Timer1, used for encoder.

## Returns

void

## 12.1.2.16 static void timer3ISR ( void ) [static]

ISR callback function for Timer3, used for pulse feature.

## Returns

void

### 12.1.3 Variable Documentation

#### 12.1.3.1 aDCEngine\* pThis = NULL

## 12.2 aDCLoad.h File Reference

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

### Classes

- class [aStepper](#)  
*Class that handles increase/decrease step multiplier.*
- class [aDCSettings](#)  
*Class that handles settings.*
- struct [aDCSettings::CalibrationData\\_t](#)  
*Calibration values.*
- union [aDCSettings::\\_eepromCalibrationValue\\_t](#)  
*Union to manipulate float/uint8\_t [] calibration values.*
- class [aLCD](#)  
*LiquidDisplay extension class.*
- class [aDCDisplay](#)  
*Class that handles LCD displaying.*
- class [aInputRelay](#)  
*Class that handles input relay.*
- class [aDCEngine](#)  
*Main class.*

### Macros

- #define [MAX\\_POWER](#) 1  
*Define this if you want 192W support (otherwise 50W)*
- #define [HAS\\_INPUT\\_RELAY](#) 1  
*Define this if you want input relay support.*
- #define [HAS\\_TWIN\\_MOSFET](#) 1  
*Define this if you're using two BUK MosFETs.*

### Variables

- static const uint8\_t [ADC\\_CHIPSELECT\\_PIN](#) = 8  
*set pin 8 as the chip select for the ADC:*
- static const uint8\_t [ADC\\_INPUTVOLTAGE\\_CHAN](#) = 0  
*set the ADC channel that reads the input voltage.*
- static const uint8\_t [ADC\\_MEASUREDCURRENT\\_CHAN](#) = 1  
*set the ADC channel that reads the input current by measuring the voltage on the input side of the sense resistors.*
- static const uint8\_t [ADC\\_TEMPSENSE1\\_CHAN](#) = 2  
*set the ADC channel that reads the temperature sensor 1 under the heatsink.*
- static const uint8\_t [ADC\\_TEMPSENSE2\\_CHAN](#) = 3

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

  - static const uint8\_t `DAC_CHIPSELECT_PIN` = 9

*set pin 9 as the chip select for the DAC:*
- static const uint8\_t `DAC_CURRENT_CHAN` = 0

*set The DAC channel that sets the constant current.*
- static const uint8\_t `DAC_FAN_CHAN` = 1

*set The DAC channel that sets the fan speed.*
- static const uint8\_t `LCD_RS_PIN` = 10

*LCD RS pin.*
- static const uint8\_t `LCD_ENABLE_PIN` = 12

*LCD ENABLE pin.*
- static const uint8\_t `LCD_D4_PIN` = A0

*LCD d4 pin.*
- static const uint8\_t `LCD_D5_PIN` = A1

*LCD d5 pin.*
- static const uint8\_t `LCD_D6_PIN` = A2

*LCD d6 pin.*
- static const uint8\_t `LCD_D7_PIN` = A3

*LCD d7 pin.*
- static const uint8\_t `LCD_COLS_NUM` = 20

*LCD columns size.*
- static const uint8\_t `LCD_ROWS_NUM` = 4

*LCD rows size.*
- static const uint8\_t `INPUT_RELAY_BUTTON_PIN` = 13

*Input Relay command button.*
- static const uint8\_t `INPUT_RELAY_RELAY_PIN` = 4

*Input Relay command pin.*
- static const uint8\_t `ENCODER_A_PIN` = 3

*Encoder Channel A pin, INT 0.*
- static const uint8\_t `ENCODER_B_PIN` = 2

*Encoder Channel B pin, INT 1.*
- static const uint8\_t `ENCODER_PB_PIN` = 0

*Encoder push button pin, INT 2.*
- static const uint8\_t `ENCODER_STEPS_PER_NOTCH` = 4

*Depending on the type of your encoder, you can define use the constructors parameter `stepsPerNotch` an set it to either 1, 2 or 4 steps per notch, with 1 being the default.*
- static const uint8\_t `LED_BACKLIGHT_PIN` = 11

*LCD backlight pin.*
- static const uint8\_t `OFFSET_UNIT` = 1

*Unit column LCD offset.*
- static const uint8\_t `OFFSET_VALUE` = 4

*Value column LCD offset.*
- static const uint8\_t `OFFSET_TEMP` = 12

*Temperature column LCD offset.*
- static const uint8\_t `OFFSET_MARKER_LEFT` = 0

*Column LCD offset for left marker '['.*
- static const uint8\_t `OFFSET_MARKER_RIGHT` = 14

*Column LCD offset for right marker ']'.*
- static const uint8\_t `OFFSET_SETUP_MARKER_LEFT` = 1

*Column LCD offset for left marker '[' in setup mode.*
- static const uint8\_t `OFFSET_SETUP_MARKER_RIGHT` = 18

- Column LCD offset for right marker 'J' in setup mode.*

  - static const uint8\_t **LOGGING\_ICON\_X\_COORD** = 17  
*Logging icon LCD X coord.*
  - static const uint8\_t **LOGGING\_ICON\_Y\_COORD** = 3  
*Logging icon LCD Y coord.*
  - static const uint8\_t **USB\_ICON\_X\_COORD** = 18  
*USB icon LCD X coord.*
  - static const uint8\_t **USB\_ICON\_Y\_COORD** = 3  
*USB icon LCD Y coord.*
  - static const uint8\_t **LOCK\_ICON\_X\_COORD** = 19  
*LOCK icon LCD X coord.*
  - static const uint8\_t **LOCK\_ICON\_Y\_COORD** = 3  
*LOCK icon LCD Y coord.*
  - static const uint8\_t **ALARM\_OV\_X\_COORD** = 18  
*Over-voltage 'OV' text LCD X coord.*
  - static const uint8\_t **ALARM\_OV\_Y\_COORD** = 1  
*Over-voltage 'OV' text LCD Y coord.*
  - static const uint8\_t **ALARM\_OC\_X\_COORD** = 18  
*Over-current 'OC' text LCD X coord.*
  - static const uint8\_t **ALARM\_OC\_Y\_COORD** = 2  
*Over-current 'OC' text LCD Y coord.*
  - static const uint8\_t **ALARM\_OT\_X\_COORD** = 18  
*Over-temperature 'OT' text LCD X coord.*
  - static const uint8\_t **ALARM\_OT\_Y\_COORD** = 2  
*Over-temperature 'OT' text LCD Y coord.*
  - static const unsigned long **AUTOLOCK\_TIMEOUT** = 60000  
*Autolock timeout value (60 seconds)*
  - static const unsigned long **OPERATION\_SET\_TIMEOUT** = 3000  
*Automatic toggle settings-> reading timeout (3 seconds)*
  - static const unsigned long **BACKLIGHT\_TIMEOUT** = 600000  
*Backlight dimmer timeout (10 minutes)*
  - static const unsigned long **LOGGING\_RATE** = 100  
*CSV data-logging rate (ms)*
  - static const unsigned long **DISPLAY\_UPDATE\_RATE** = 200  
*Display update rate (ms)*
  - static const float **VOLTAGE\_MAXIMUM** = 24.000  
*Maximum handled voltage (V)*
  - static const float **CURRENT\_MAXIMUM** = 7.845  
*Maximum value of load current (A)*
  - static const float **POWER\_MAXIMUM** = **VOLTAGE\_MAXIMUM** \* **CURRENT\_MAXIMUM**  
*Maximum power dissipated (W)*
  - static const float **PULSE\_MAXIMUM** = 8.192  
*Maximum pulse value (ms) (unsigned 16-bit)*
  - static const uint16\_t **TEMPERATURE\_MAXIMUM** = 80  
*Over-temperature threshold.*
  - static const int8\_t **SOFTWARE\_VERSION\_MAJOR** = 2  
*Software major version.*
  - static const int8\_t **SOFTWARE\_VERSION\_MINOR** = 6  
*Software minor version.*
  - static const uint16\_t **FEATURE\_LOGGING** = 1  
*Bitfield logging feature.*



- static const uint16\_t [FEATURE\\_LOGGING\\_VISIBLE](#) = 1 << 1  
*Bitfield logging icon feature.*
- static const uint16\_t [FEATURE\\_USB](#) = 1 << 2  
*Bitfield USB feature.*
- static const uint16\_t [FEATURE\\_USB\\_VISIBLE](#) = 1 << 3  
*Bitfield USB icon feature.*
- static const uint16\_t [FEATURE\\_LOCKED](#) = 1 << 4  
*Bitfield locking feature.*
- static const uint16\_t [FEATURE\\_LOCKED\\_VISIBLE](#) = 1 << 5  
*Bitfield locking icon feature.*
- static const uint16\_t [FEATURE\\_AUTOLOCK](#) = 1 << 6  
*Bitfield auto-lock setting feature.*
- static const uint16\_t [FEATURE\\_AUTOLOCK\\_VISIBLE](#) = 1 << 7  
*Bitfield auto-lock setting icon feature.*
- static const uint16\_t [FEATURE\\_AUTODIM](#) = 1 << 8  
*Bitfield auto-dimmer setting feature.*
- static const uint16\_t [FEATURE\\_AUTODIM\\_VISIBLE](#) = 1 << 9  
*Bitfield auto-dimmer setting icon feature.*
- static const uint16\_t [FEATURE\\_OVP](#) = 1 << 10  
*Bitfield over-voltage protection feature.*
- static const uint16\_t [FEATURE\\_OVP\\_VISIBLE](#) = 1 << 11  
*Bitfield over-voltage protection icon feature.*
- static const uint16\_t [FEATURE\\_OCP](#) = 1 << 12  
*Bitfield over-current protection feature.*
- static const uint16\_t [FEATURE\\_OCP\\_VISIBLE](#) = 1 << 13  
*Bitfield over-current protection icon feature.*
- static const uint16\_t [FEATURE\\_OTP](#) = 1 << 14  
*Bitfield over-temperature protection feature.*
- static const uint16\_t [FEATURE\\_OTP\\_VISIBLE](#) = 1 << 15  
*Bitfield over-temperature protection icon feature.*
- static const uint8\_t [GLYPH\\_X1](#) = 0  
*Offset of \*1 icon in LCD HD44780 controller memory.*
- static const uint8\_t [GLYPH\\_X10](#) = 1  
*Offset of \*10 icon in LCD HD44780 controller memory.*
- static const uint8\_t [GLYPH\\_X100](#) = 2  
*Offset of \*100 icon in LCD HD44780 controller memory.*
- static const uint8\_t [GLYPH\\_X1000](#) = 3  
*Offset of \*1000 icon in LCD HD44780 controller memory.*
- static const uint8\_t [GLYPH\\_USB](#) = 4  
*Offset of USB icon in LCD HD44780 controller memory.*
- static const uint8\_t [GLYPH\\_LOCK](#) = 5  
*Offset of LOCK icon in LCD HD44780 controller memory.*
- static const uint8\_t [GLYPH\\_CHECKBOX\\_UNTICKED](#) = 6  
*Offset of unticked checkbox icon in LCD HD44780 controller memory.*
- static const uint8\_t [GLYPH\\_CHECKBOX\\_TICKED](#) = 7  
*Offset of ticked checkbox icon in LCD HD44780 controller memory.*
- static const int16\_t [EEPROM\\_ADDR\\_MAGIC](#) = 0  
*EEPROM start offset to magic numbers (0xDEAD)*
- static const int16\_t [EEPROM\\_ADDR\\_AUTODIM](#) = 4  
*EEPROM start offset for autodimming setting.*
- static const int16\_t [EEPROM\\_ADDR\\_AUTOLOCK](#) = 5

- EEPROM start offset for autolocking setting.*

• static const int16\_t `EEPROM_CALIBRATION_SIZE` = (sizeof(float) \* 2) + sizeof(uint8\_t)

*EEPROM calibration size: 2 float (slope & offset), and one uint8\_t for crc.*
- static const int16\_t `EEPROM_ADDR_CALIBRATION_VOLTAGE` = `EEPROM_ADDR_AUTOLOCK` + 1

*EEPROM start offset for voltage calibration values.*
- static const int16\_t `EEPROM_ADDR_CALIBRATION_READ_CURRENT` = `EEPROM_ADDR_CALIBRATION_VOLTAGE` + `EEPROM_CALIBRATION_SIZE`

*EEPROM start offset for current calibration values.*
- static const int16\_t `EEPROM_ADDR_CALIBRATION_DAC_CURRENT` = `EEPROM_ADDR_CALIBRATION_READ_CURRENT` + `EEPROM_CALIBRATION_SIZE`

*EEPROM start offset for DAC calibration values.*
- static const int16\_t `EEPROM_ADDR_CALIBRATION_VOLTAGE_DROP` = `EEPROM_ADDR_CALIBRATION_DAC_CURRENT` + `EEPROM_CALIBRATION_SIZE`

*EEPROM start offset for DAC calibration values.*

## 12.2.1 Detailed Description

### Copyright

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

## 12.2.2 Macro Definition Documentation

### 12.2.2.1 #define HAS\_INPUT\_RELAY 1

Define this if you want input relay support.

### 12.2.2.2 #define HAS\_TWIN\_MOSFET 1

Define this if you're using two BUK MosFETs.

### 12.2.2.3 #define MAX\_POWER 1

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

### 12.2.3 Variable Documentation

12.2.3.1 `const uint8_t ADC_CHIPSELECT_PIN = 8` [static]

set pin 8 as the chip select for the ADC:

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

12.2.3.2 `const uint8_t ADC_INPUTVOLTAGE_CHAN = 0` [static]

set the ADC channel that reads the input voltage.

12.2.3.3 `const uint8_t ADC_MEASUREDCURRENT_CHAN = 1` [static]

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

12.2.3.4 `const uint8_t ADC_TEMPSENSE1_CHAN = 2` [static]

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

12.2.3.5 `const uint8_t ADC_TEMPSENSE2_CHAN = 3` [static]

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

12.2.3.6 `const uint8_t ALARM_OC_X_COORD = 18` [static]

Over-current 'OC' text LCD X coord.

12.2.3.7 `const uint8_t ALARM_OC_Y_COORD = 2` [static]

Over-current 'OC' text LCD Y coord.

12.2.3.8 `const uint8_t ALARM_OT_X_COORD = 18` [static]

Over-temperature 'OT' text LCD X coord.

12.2.3.9 `const uint8_t ALARM_OT_Y_COORD = 2` [static]

Over-temperature 'OT' text LCD Y coord.

12.2.3.10 `const uint8_t ALARM_OV_X_COORD = 18` [static]

Over-voltage 'OV' text LCD X coord.

12.2.3.11 `const uint8_t ALARM_OV_Y_COORD = 1` [static]

Over-voltage 'OV' text LCD Y coord.

**12.2.3.12** `const unsigned long AUTOLOCK_TIMEOUT = 60000` `[static]`

Autolock timeout value (60 seconds)

**12.2.3.13** `const unsigned long BACKLIGHT_TIMEOUT = 600000` `[static]`

Backlight dimmer timeout (10 minutes)

**12.2.3.14** `const float CURRENT_MAXIMUM = 7.845` `[static]`

Maximum value of load current (A)

**12.2.3.15** `const uint8_t DAC_CHIPSELECT_PIN = 9` `[static]`

set pin 9 as the chip select for the DAC:

**12.2.3.16** `const uint8_t DAC_CURRENT_CHAN = 0` `[static]`

set The DAC channel that sets the constant current.

**12.2.3.17** `const uint8_t DAC_FAN_CHAN = 1` `[static]`

set The DAC channel that sets the fan speed.

**12.2.3.18** `const unsigned long DISPLAY_UPDATE_RATE = 200` `[static]`

Display update rate (ms)

**12.2.3.19** `const int16_t EEPROM_ADDR_AUTODIM = 4` `[static]`

EEPROM start offset for autodimming setting.

**12.2.3.20** `const int16_t EEPROM_ADDR_AUTOLOCK = 5` `[static]`

EEPROM start offset for autolocking setting.

**12.2.3.21** `const int16_t EEPROM_ADDR_CALIBRATION_DAC_CURRENT = EEPROM_ADDR_CALIBRATION_READ_CURRENT + EEPROM_CALIBRATION_SIZE` `[static]`

EEPROM start offset for DAC calibration values.

**12.2.3.22** `const int16_t EEPROM_ADDR_CALIBRATION_READ_CURRENT = EEPROM_ADDR_CALIBRATION_VOLTAGE + EEPROM_CALIBRATION_SIZE` `[static]`

EEPROM start offset for current calibration values.

**12.2.3.23** `const int16_t EEPROM_ADDR_CALIBRATION_VOLTAGE = EEPROM_ADDR_AUTOLOCK + 1` `[static]`

EEPROM start offset for voltage calibration values.

**12.2.3.24** `const int16_t EEPROM_ADDR_CALIBRATION_VOLTAGE_DROP = EEPROM_ADDR_CALIBRATION_DAC_CURRENT + EEPROM_CALIBRATION_SIZE` `[static]`

EEPROM start offset for DAC calibration values.

**12.2.3.25** `const int16_t EEPROM_ADDR_MAGIC = 0` `[static]`

EEPROM start offset to magic numbers (0xDEAD)

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

EEPROM calibration size: 2 float (slope & offset), and one uint8\_t for crc.

**12.2.3.27** `const uint8_t ENCODER_A_PIN = 3` `[static]`

Encoder Channel A pin, INT 0.

**12.2.3.28** `const uint8_t ENCODER_B_PIN = 2` `[static]`

Encoder Channel B pin, INT 1.

**12.2.3.29** `const uint8_t ENCODER_PB_PIN = 0` `[static]`

Encoder push button pin, INT 2.

**12.2.3.30** `const uint8_t ENCODER_STEPS_PER_NOTCH = 4` `[static]`

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

**12.2.3.31** `const uint16_t FEATURE_AUTODIM = 1 << 8` `[static]`

Bitfield auto-dimmer setting feature.

**12.2.3.32** `const uint16_t FEATURE_AUTODIM_VISIBLE = 1 << 9` `[static]`

Bitfield auto-dimmer setting icon feature.

**12.2.3.33** `const uint16_t FEATURE_AUTOLOCK = 1 << 6` `[static]`

Bitfield auto-lock setting feature.

**12.2.3.34** `const uint16_t FEATURE_AUTOLOCK_VISIBLE = 1 << 7` `[static]`

Bitfield auto-lock setting icon feature.

**12.2.3.35** `const uint16_t FEATURE_LOCKED = 1 << 4` `[static]`

Bitfield locking feature.

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

Bitfield locking icon feature.

12.2.3.37 `const uint16_t FEATURE_LOGGING = 1 [static]`

Bitfield logging feature.

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

Bitfield logging icon feature.

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

Bitfield over-current protection feature.

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

Bitfield over-current protection icon feature.

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

Bitfield over-temperature protection feature.

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

Bitfield over-temperature protection icon feature.

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

Bitfield over-voltage protection feature.

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

Bitfield over-voltage protection icon feature.

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

Bitfield USB feature.

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

Bitfield USB icon feature.

12.2.3.47 `const uint8_t GLYPH_CHECKBOX_TICKED = 7 [static]`

Offset of ticked checkbox icon in LCD HD44780 controller memory.

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

Offset of unticked checkbox icon in LCD HD44780 controller memory.

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

Offset of LOCK icon in LCD HD44780 controller memory.

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

Offset of USB icon in LCD HD44780 controller memory.

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

Offset of \*1 icon in LCD HD44780 controller memory.

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

Offset of \*10 icon in LCD HD44780 controller memory.

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

Offset of \*100 icon in LCD HD44780 controller memory.

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

Offset of \*1000 icon in LCD HD44780 controller memory.

12.2.3.55 `const uint8_t INPUT_RELAY_BUTTON_PIN = 13` `[static]`

Input Relay command button.

12.2.3.56 `const uint8_t INPUT_RELAY_RELAY_PIN = 4` `[static]`

Input Relay command pin.

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

LCD columns size.

12.2.3.58 `const uint8_t LCD_D4_PIN = A0` `[static]`

LCD d4 pin.

#### Warning

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

12.2.3.59 `const uint8_t LCD_D5_PIN = A1` `[static]`

LCD d5 pin.

**Warning**

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

12.2.3.60 `const uint8_t LCD_D6_PIN = A2` `[static]`

LCD d6 pin.

**Warning**

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

12.2.3.61 `const uint8_t LCD_D7_PIN = A3` `[static]`

LCD d7 pin.

**Warning**

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

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

LCD ENABLE pin.

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

LCD rows size.

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

LCD RS pin.

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

LCD backlight pin.

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

LOCK icon LCD X coord.

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

LOCK icon LCD Y coord.

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

Logging icon LCD X coord.



**12.2.3.69** `const uint8_t LOGGING_ICON_Y_COORD = 3` `[static]`

Logging icon LCD Y coord.

**12.2.3.70** `const unsigned long LOGGING_RATE = 100` `[static]`

CSV data-logging rate (ms)

**12.2.3.71** `const uint8_t OFFSET_MARKER_LEFT = 0` `[static]`

Column LCD offset for left marker '['.

**12.2.3.72** `const uint8_t OFFSET_MARKER_RIGHT = 14` `[static]`

Column LCD offset for right marker ']'.

**12.2.3.73** `const uint8_t OFFSET_SETUP_MARKER_LEFT = 1` `[static]`

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

**12.2.3.74** `const uint8_t OFFSET_SETUP_MARKER_RIGHT = 18` `[static]`

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

**12.2.3.75** `const uint8_t OFFSET_TEMP = 12` `[static]`

Temperature column LCD offset.

**12.2.3.76** `const uint8_t OFFSET_UNIT = 1` `[static]`

Unit column LCD offset.

**12.2.3.77** `const uint8_t OFFSET_VALUE = 4` `[static]`

Value column LCD offset.

**12.2.3.78** `const unsigned long OPERATION_SET_TIMEOUT = 3000` `[static]`

Automatic toggle settings->reading timeout (3 seconds)

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

Maximum power dissipated (W)

**12.2.3.80** `const float PULSE_MAXIMUM = 8.192` `[static]`

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

**12.2.3.81** `const int8_t SOFTWARE_VERSION_MAJOR = 2` [static]

Software major version.

**12.2.3.82** `const int8_t SOFTWARE_VERSION_MINOR = 6` [static]

Software minor version.

**12.2.3.83** `const uint16_t TEMPERATURE_MAXIMUM = 80` [static]

Over-temperature threshold.

**12.2.3.84** `const uint8_t USB_ICON_X_COORD = 18` [static]

USB icon LCD X coord.

**12.2.3.85** `const uint8_t USB_ICON_Y_COORD = 3` [static]

USB icon LCD Y coord.

**12.2.3.86** `const float VOLTAGE_MAXIMUM = 24.000` [static]

Maximum handled voltage (V)

## 12.3 CDC.cpp File Reference

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

## 12.4 HardwareSerial.cpp File Reference

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

## 12.5 HID.cpp File Reference

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

## 12.6 IPAddress.cpp File Reference

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

## 12.7 libraries.cpp File Reference

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

## 12.8 main.cpp File Reference

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

## 12.9 new.cpp File Reference

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

## 12.10 Print.cpp File Reference

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

## 12.11 sketch.cpp File Reference

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

### Functions

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

### Variables

- [aDCEngine engine](#) (LCD\_RS\_PIN, LCD\_ENABLE\_PIN, LCD\_D4\_PIN, LCD\_D5\_PIN, LCD\_D6\_PIN, LCD\_D7\_PIN, LCD\_COLS\_NUM, LCD\_ROWS\_NUM, INPUT\_RELAY\_BUTTON\_PIN, INPUT\_RELAY\_RELAY\_PIN, ENCODER\_A\_PIN, ENCODER\_B\_PIN, ENCODER\_PB\_PIN, ENCODER\_STEPS\_PER\_NOTCH)

### 12.11.1 Function Documentation

12.11.1.1 void [loop](#) ( )

12.11.1.2 void [setup](#) ( )

### 12.11.2 Variable Documentation

12.11.2.1 [aDCEngine engine](#)(LCD\_RS\_PIN, LCD\_ENABLE\_PIN, LCD\_D4\_PIN, LCD\_D5\_PIN, LCD\_D6\_PIN, LCD\_D7\_PIN, LCD\_COLS\_NUM, LCD\_ROWS\_NUM, INPUT\_RELAY\_BUTTON\_PIN, INPUT\_RELAY\_RELAY\_PIN, ENCODER\_A\_PIN, ENCODER\_B\_PIN, ENCODER\_PB\_PIN, ENCODER\_STEPS\_PER\_NOTCH)

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

## 12.12 Stream.cpp File Reference

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

### 12.13 Tone.cpp File Reference

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

### 12.14 USBCore.cpp File Reference

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

### 12.15 WInterrupts.c File Reference

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

### 12.16 wiring.c File Reference

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

### 12.17 wiring\_analog.c File Reference

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

### 12.18 wiring\_digital.c File Reference

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

### 12.19 wiring\_pulse.c File Reference

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

### 12.20 wiring\_shift.c File Reference

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

### 12.21 WMath.cpp File Reference

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

### 12.22 WString.cpp File Reference

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

# Index

- ~aDCDisplay
  - aDCDisplay, [29](#)
- ~aDCEngine
  - aDCEngine, [33](#)
- ~aDCSettings
  - aDCSettings, [42](#)
- ~aInputRelay
  - aInputRelay, [57](#)
- ~aLCD
  - aLCD, [59](#)
- ~aStepper
  - aStepper, [62](#)
- \_adjustLoadCurrent
  - aDCEngine, [33](#)
- \_crc8
  - aDCSettings, [42](#)
- \_dimBacklight
  - aDCDisplay, [29](#)
- \_dimmingBacklight
  - aDCDisplay, [29](#)
- \_eepromCalibrationBackup
  - aDCSettings, [42](#)
- \_eepromCalibrationRestore
  - aDCSettings, [42](#)
- \_eepromCheckForMagicNumbers
  - aDCSettings, [42](#)
- \_eepromReset
  - aDCSettings, [43](#)
- \_eepromRestore
  - aDCSettings, [43](#)
- \_eepromWriteMagicNumbers
  - aDCSettings, [43](#)
- \_enableData
  - aDCSettings, [43](#)
- \_enableDataCheck
  - aDCSettings, [43](#)
- \_getADC
  - aDCEngine, [33](#)
- \_getInputVoltage
  - aDCEngine, [33](#)
- \_getMeasuredCurrent
  - aDCEngine, [33](#)
- \_getTemp
  - aDCEngine, [33](#)
- \_handleButtonEvent
  - aDCEngine, [34](#)
- \_handleLoggingAndRemote
  - aDCEngine, [34](#)
- \_pow

- aStepper, [62](#)
- \_setDAC
  - aDCEngine, [34](#)
- \_setValue
  - aDCSettings, [44](#)
- \_updateFanSpeed
  - aDCEngine, [34](#)
- \_updateLoadCurrent
  - aDCEngine, [34](#)
- \_wakeupBacklight
  - aDCDisplay, [29](#)
- aDCSettings
  - CALIBRATION\_DAC\_CURRENT, [40](#)
  - CALIBRATION\_MAX, [40](#)
  - CALIBRATION\_READ\_CURRENT, [40](#)
  - CALIBRATION\_VOLTAGE, [40](#)
  - CALIBRATION\_VOLTAGE\_DROP, [40](#)
  - DISPLAY\_MODE\_SETUP, [41](#)
  - DISPLAY\_MODE\_UNKNOWN, [41](#)
  - DISPLAY\_MODE\_VALUES, [41](#)
  - OPERATION\_MODE\_READ, [41](#)
  - OPERATION\_MODE\_SET, [41](#)
  - OPERATION\_MODE\_UNKNOWN, [41](#)
  - SELECTION\_MODE\_CURRENT, [41](#)
  - SELECTION\_MODE\_POWER, [41](#)
  - SELECTION\_MODE\_PULSE, [41](#)
  - SELECTION\_MODE\_UNKNOWN, [41](#)
  - SETTING\_ERROR\_OVERSIZED, [41](#)
  - SETTING\_ERROR\_UNDERSIZED, [41](#)
  - SETTING\_ERROR\_VALID, [41](#)
- ADC\_CHIPSELECT\_PIN
  - aDCLoad.h, [77](#)
- ADC\_TEMPSENSE1\_CHAN
  - aDCLoad.h, [77](#)
- ADC\_TEMPSENSE2\_CHAN
  - aDCLoad.h, [77](#)
- aDCDisplay, [27](#)
  - ~aDCDisplay, [29](#)
  - \_dimBacklight, [29](#)
  - \_dimmingBacklight, [29](#)
  - \_wakeupBacklight, [29](#)
  - aDCDisplay, [28](#)
  - aDCDisplay, [28](#)
  - aDCEngine, [35](#)
  - isBacklightDimmed, [29](#)
  - m\_Parent, [31](#)
  - m\_dimmed, [30](#)
  - m\_dimmerTick, [30](#)
  - m\_nextUpdate, [31](#)

- pingBacklight, 29
- setup, 30
- showBanner, 30
- updateDisplay, 30
- updateField, 30
- aDCEngine, 31
  - ~aDCEngine, 33
  - \_adjustLoadCurrent, 33
  - \_getADC, 33
  - \_getInputVoltage, 33
  - \_getMeasuredCurrent, 33
  - \_getTemp, 33
  - \_handleButtonEvent, 34
  - \_handleLoggingAndRemote, 34
  - \_setDAC, 34
  - \_updateFanSpeed, 34
  - \_updateLoadCurrent, 34
- aDCDisplay, 35
- aDCEngine, 32
- aDCEngine, 32
- getEncoder, 34
- getSettings, 35
- m\_Data, 35
- m\_RXbuffer, 35
- m\_RXoffset, 35
- m\_encoder, 35
- m\_inputRelay, 35
- RXBUFFER\_MAXLEN, 36
- run, 35
- setInput, 35
- setup, 35
- aDCLoad.cpp, 67
  - floatRounding, 68
  - getNumericalLength, 69
  - pThis, 72
  - serialAvailable, 69
  - serialFlush, 69
  - serialPrint, 69, 70
  - serialPrintln, 70, 71
  - serialRead, 71
  - serialWrite, 71
  - timer1ISR, 71
  - timer3ISR, 71
- aDCLoad.h, 72
  - AUTOLOCK\_TIMEOUT, 77
  - BACKLIGHT\_TIMEOUT, 78
  - CURRENT\_MAXIMUM, 78
  - DAC\_CURRENT\_CHAN, 78
  - DAC\_FAN\_CHAN, 78
  - ENCODER\_A\_PIN, 79
  - ENCODER\_B\_PIN, 79
  - ENCODER\_PB\_PIN, 79
  - FEATURE\_AUTODIM, 79
  - FEATURE\_AUTOLOCK, 79
  - FEATURE\_LOCKED, 79
  - FEATURE\_LOGGING, 80
  - FEATURE\_OCP, 80
  - FEATURE\_OTP, 80
  - FEATURE\_OVP, 80
  - FEATURE\_USB, 80
  - GLYPH\_LOCK, 81
  - GLYPH\_USB, 81
  - GLYPH\_X1, 81
  - GLYPH\_X10, 81
  - GLYPH\_X100, 81
  - GLYPH\_X1000, 81
  - HAS\_INPUT\_RELAY, 76
  - HAS\_TWIN\_MOSFET, 76
  - LCD\_COLS\_NUM, 81
  - LCD\_D4\_PIN, 81
  - LCD\_D5\_PIN, 81
  - LCD\_D6\_PIN, 82
  - LCD\_D7\_PIN, 82
  - LCD\_ENABLE\_PIN, 82
  - LCD\_ROWS\_NUM, 82
  - LCD\_RS\_PIN, 82
  - LOGGING\_RATE, 83
  - MAX\_POWER, 76
  - OFFSET\_TEMP, 83
  - OFFSET\_UNIT, 83
  - OFFSET\_VALUE, 83
  - POWER\_MAXIMUM, 83
  - PULSE\_MAXIMUM, 83
  - VOLTAGE\_MAXIMUM, 84
- aDCSettings, 36
  - ~aDCSettings, 42
  - \_crc8, 42
  - \_eepromCalibrationBackup, 42
  - \_eepromCalibrationRestore, 42
  - \_eepromCheckForMagicNumbers, 42
  - \_eepromReset, 43
  - \_eepromRestore, 43
  - \_eepromWriteMagicNumbers, 43
  - \_enableData, 43
  - \_enableDataCheck, 43
  - \_setValue, 44
- aDCSettings, 41
- aDCSettings, 41
  - backupCalibration, 44
  - CalibrationValues\_t, 40
  - DATA\_CURRENT\_READ, 54
  - DATA\_CURRENT\_SETS, 54
  - DATA\_DISPLAY, 54
  - DATA\_ENCODER, 54
  - DATA\_OPERATION, 54
  - DATA\_POWER\_READ, 54
  - DATA\_POWER\_SETS, 54
  - DATA\_PULSE\_SETS, 54
  - DATA\_SELECTION, 54
  - DATA\_TEMPERATURE, 54
  - DATA\_VOLTAGE, 54
  - DisplayMode\_t, 40
  - enableAlarm, 44
  - enableFeature, 44
  - enablePulse, 45
  - getCalibrationMode, 45

- getCalibrationValues, 45
- getCurrent, 45
- getCurrentDAC, 45
- getDisplayMode, 46
- getEncoderPosition, 46
- getFanSpeed, 46
- getOperationMode, 46
- getPower, 46
- getPrevNextMode, 46
- getPulse, 47
- getSelectionMode, 47
- getTemperature, 47
- getVoltage, 47
- incEncoderPosition, 47
- isAutolocked, 47
- isDataEnabled, 48
- isFeatureEnabled, 48
- isPulseEnabled, 48
- isPulseHigh, 48
- m\_Parent, 56
- m\_calibrationValues, 54
- m\_currentDAC, 55
- m\_datas, 55
- m\_dispMode, 55
- m\_encoderPos, 55
- m\_fanSpeed, 55
- m\_features, 55
- m\_lockTick, 55
- m\_mode, 55
- m\_operationMode, 55
- m\_operationTick, 55
- m\_pulseEnabled, 56
- m\_pulseHigh, 56
- m\_readCurrent, 56
- m\_readPower, 56
- m\_readTemperature, 56
- m\_readVoltage, 56
- m\_setsCurrent, 56
- m\_setsPower, 56
- m\_setsPulse, 56
- OperationMode\_t, 41
- pingAutolock, 48
- pingOperationMode, 48
- restoreCalibration, 49
- SelectionMode\_t, 41
- setCalibrationMode, 49
- setCalibrationValues, 49
- setCurrent, 49
- setCurrentDAC, 49
- setDisplayMode, 50
- setEncoderPosition, 50
- setFanSpeed, 50
- setOperationMode, 50
- setPower, 50
- setPulse, 52
- setPulseHigh, 52
- setSelectionMode, 52
- setTemperature, 52
- setVoltage, 53
- SettingError\_t, 41
- syncData, 53
- updateOperationMode, 53
- updateValuesFromMode, 53
- aDCSettings::\_eepromCalibrationValue\_t, 27
  - c, 27
  - v, 27
- aDCSettings::CalibrationData\_t, 64
  - offset, 64
  - slope, 64
- aInputRelay, 56
  - ~aInputRelay, 57
  - aInputRelay, 57
  - aInputRelay, 57
  - isClicked, 57
  - isInput, 57
  - m\_btnPin, 58
  - m\_btnState, 58
  - m\_clicked, 58
  - m\_isON, 58
  - m\_relayPin, 58
  - service, 58
  - setInput, 58
- ALARM\_OC\_X\_COORD
  - aDCLoad.h, 77
- ALARM\_OC\_Y\_COORD
  - aDCLoad.h, 77
- ALARM\_OT\_X\_COORD
  - aDCLoad.h, 77
- ALARM\_OT\_Y\_COORD
  - aDCLoad.h, 77
- ALARM\_OV\_X\_COORD
  - aDCLoad.h, 77
- ALARM\_OV\_Y\_COORD
  - aDCLoad.h, 77
- aLCD, 58
  - ~aLCD, 59
  - aLCD, 59
  - aLCD, 59
  - begin, 60
  - clearValue, 60
  - m\_cols, 61
  - m\_curCol, 61
  - m\_curRow, 61
  - m\_rows, 61
  - printCenter, 60
  - setCursor, 60
- aStepper, 61
  - ~aStepper, 62
  - \_pow, 62
  - aStepper, 62
  - aStepper, 62
  - getMult, 62
  - getValue, 63
  - getValueFromMode, 63
  - increment, 63
  - isSynced, 63

- m\_inc, 64
  - m\_incPrev, 64
  - MAX\_VALUE, 64
  - reset, 63
  - sync, 63
- AUTOLOCK\_TIMEOUT
  - aDCLoad.h, 77
- BACKLIGHT\_TIMEOUT
  - aDCLoad.h, 78
- backupCalibration
  - aDCSettings, 44
- begin
  - aLCD, 60
- c
  - aDCSettings::\_eepromCalibrationValue\_t, 27
- CALIBRATION\_DAC\_CURRENT
  - aDCSettings, 40
- CALIBRATION\_MAX
  - aDCSettings, 40
- CALIBRATION\_READ\_CURRENT
  - aDCSettings, 40
- CALIBRATION\_VOLTAGE
  - aDCSettings, 40
- CALIBRATION\_VOLTAGE\_DROP
  - aDCSettings, 40
- CDC.cpp, 84
- CURRENT\_MAXIMUM
  - aDCLoad.h, 78
- CalibrationValues\_t
  - aDCSettings, 40
- clearValue
  - aLCD, 60
- DISPLAY\_MODE\_SETUP
  - aDCSettings, 41
- DISPLAY\_MODE\_UNKNOWN
  - aDCSettings, 41
- DISPLAY\_MODE\_VALUES
  - aDCSettings, 41
- DAC\_CHIPSELECT\_PIN
  - aDCLoad.h, 78
- DAC\_CURRENT\_CHAN
  - aDCLoad.h, 78
- DAC\_FAN\_CHAN
  - aDCLoad.h, 78
- DATA\_CURRENT\_READ
  - aDCSettings, 54
- DATA\_CURRENT\_SETS
  - aDCSettings, 54
- DATA\_DISPLAY
  - aDCSettings, 54
- DATA\_ENCODER
  - aDCSettings, 54
- DATA\_OPERATION
  - aDCSettings, 54
- DATA\_POWER\_READ
  - aDCSettings, 54
- DATA\_POWER\_SETS
  - aDCSettings, 54
- DATA\_PULSE\_SETS
  - aDCSettings, 54
- DATA\_SELECTION
  - aDCSettings, 54
- DATA\_TEMPERATURE
  - aDCSettings, 54
- DATA\_VOLTAGE
  - aDCSettings, 54
- DisplayMode\_t
  - aDCSettings, 40
- EEPROM\_ADDR\_MAGIC
  - aDCLoad.h, 79
- ENCODER\_A\_PIN
  - aDCLoad.h, 79
- ENCODER\_B\_PIN
  - aDCLoad.h, 79
- ENCODER\_PB\_PIN
  - aDCLoad.h, 79
- enableAlarm
  - aDCSettings, 44
- enableFeature
  - aDCSettings, 44
- enablePulse
  - aDCSettings, 45
- engine
  - sketch.cpp, 85
- FEATURE\_AUTODIM
  - aDCLoad.h, 79
- FEATURE\_AUTOLOCK
  - aDCLoad.h, 79
- FEATURE\_LOCKED
  - aDCLoad.h, 79
- FEATURE\_LOGGING
  - aDCLoad.h, 80
- FEATURE\_OCP
  - aDCLoad.h, 80
- FEATURE\_OTP
  - aDCLoad.h, 80
- FEATURE\_OVP
  - aDCLoad.h, 80
- FEATURE\_USB
  - aDCLoad.h, 80
- floatRounding
  - aDCLoad.cpp, 68
- GLYPH\_LOCK
  - aDCLoad.h, 81
- GLYPH\_USB
  - aDCLoad.h, 81
- GLYPH\_X1
  - aDCLoad.h, 81
- GLYPH\_X10
  - aDCLoad.h, 81
- GLYPH\_X100
  - aDCLoad.h, 81



- GLYPH\_X1000
  - aDCLoad.h, 81
- getCalibrationMode
  - aDCSettings, 45
- getCalibrationValues
  - aDCSettings, 45
- getCurrent
  - aDCSettings, 45
- getCurrentDAC
  - aDCSettings, 45
- getDisplayMode
  - aDCSettings, 46
- getEncoder
  - aDCEngine, 34
- getEncoderPosition
  - aDCSettings, 46
- getFanSpeed
  - aDCSettings, 46
- getMult
  - aStepper, 62
- getNumericalLength
  - aDCLoad.cpp, 69
- getOperationMode
  - aDCSettings, 46
- getPower
  - aDCSettings, 46
- getPrevNextMode
  - aDCSettings, 46
- getPulse
  - aDCSettings, 47
- getSelectionMode
  - aDCSettings, 47
- getSettings
  - aDCEngine, 35
- getTemperature
  - aDCSettings, 47
- getValue
  - aStepper, 63
- getValueFromMode
  - aStepper, 63
- getVoltage
  - aDCSettings, 47
  
- HAS\_INPUT\_RELAY
  - aDCLoad.h, 76
- HAS\_TWIN\_MOSFET
  - aDCLoad.h, 76
- HID.cpp, 84
- HardwareSerial.cpp, 84
  
- IPAddress.cpp, 84
- incEncoderPosition
  - aDCSettings, 47
- increment
  - aStepper, 63
- isAutolocked
  - aDCSettings, 47
- isBacklightDimmed
  - aDCDisplay, 29
- isClicked
  - aInputRelay, 57
- isDataEnabled
  - aDCSettings, 48
- isFeatureEnabled
  - aDCSettings, 48
- isInput
  - aInputRelay, 57
- isPulseEnabled
  - aDCSettings, 48
- isPulseHigh
  - aDCSettings, 48
- isSynced
  - aStepper, 63
  
- LCD\_COLS\_NUM
  - aDCLoad.h, 81
- LCD\_D4\_PIN
  - aDCLoad.h, 81
- LCD\_D5\_PIN
  - aDCLoad.h, 81
- LCD\_D6\_PIN
  - aDCLoad.h, 82
- LCD\_D7\_PIN
  - aDCLoad.h, 82
- LCD\_ENABLE\_PIN
  - aDCLoad.h, 82
- LCD\_ROWS\_NUM
  - aDCLoad.h, 82
- LCD\_RS\_PIN
  - aDCLoad.h, 82
- LED\_BACKLIGHT\_PIN
  - aDCLoad.h, 82
- LOCK\_ICON\_X\_COORD
  - aDCLoad.h, 82
- LOCK\_ICON\_Y\_COORD
  - aDCLoad.h, 82
- LOGGING\_RATE
  - aDCLoad.h, 83
- libraries.cpp, 84
- loop
  - sketch.cpp, 85
  
- m\_Data
  - aDCEngine, 35
- m\_Parent
  - aDCDisplay, 31
  - aDCSettings, 56
- m\_RXbuffer
  - aDCEngine, 35
- m\_RXoffset
  - aDCEngine, 35
- m\_btnPin
  - aInputRelay, 58
- m\_btnState
  - aInputRelay, 58
- m\_calibrationValues
  - aDCSettings, 54
- m\_clicked

- aInputRelay, 58
- m\_cols
  - aLCD, 61
- m\_curCol
  - aLCD, 61
- m\_curRow
  - aLCD, 61
- m\_currentDAC
  - aDCSettings, 55
- m\_datas
  - aDCSettings, 55
- m\_dimmed
  - aDCDisplay, 30
- m\_dimmerTick
  - aDCDisplay, 30
- m\_dispMode
  - aDCSettings, 55
- m\_encoder
  - aDCEngine, 35
- m\_encoderPos
  - aDCSettings, 55
- m\_fanSpeed
  - aDCSettings, 55
- m\_features
  - aDCSettings, 55
- m\_inc
  - aStepper, 64
- m\_incPrev
  - aStepper, 64
- m\_inputRelay
  - aDCEngine, 35
- m\_isON
  - aInputRelay, 58
- m\_lockTick
  - aDCSettings, 55
- m\_mode
  - aDCSettings, 55
- m\_nextUpdate
  - aDCDisplay, 31
- m\_operationMode
  - aDCSettings, 55
- m\_operationTick
  - aDCSettings, 55
- m\_pulseEnabled
  - aDCSettings, 56
- m\_pulseHigh
  - aDCSettings, 56
- m\_readCurrent
  - aDCSettings, 56
- m\_readPower
  - aDCSettings, 56
- m\_readTemperature
  - aDCSettings, 56
- m\_readVoltage
  - aDCSettings, 56
- m\_relayPin
  - aInputRelay, 58
- m\_rows
  - aLCD, 61
- m\_setsCurrent
  - aDCSettings, 56
- m\_setsPower
  - aDCSettings, 56
- m\_setsPulse
  - aDCSettings, 56
- MAX\_POWER
  - aDCLoad.h, 76
- MAX\_VALUE
  - aStepper, 64
- main.cpp, 85
- new.cpp, 85
- OPERATION\_MODE\_READ
  - aDCSettings, 41
- OPERATION\_MODE\_SET
  - aDCSettings, 41
- OPERATION\_MODE\_UNKNOWN
  - aDCSettings, 41
- OFFSET\_MARKER\_LEFT
  - aDCLoad.h, 83
- OFFSET\_TEMP
  - aDCLoad.h, 83
- OFFSET\_UNIT
  - aDCLoad.h, 83
- OFFSET\_VALUE
  - aDCLoad.h, 83
- offset
  - aDCSettings::CalibrationData\_t, 64
- OperationMode\_t
  - aDCSettings, 41
- POWER\_MAXIMUM
  - aDCLoad.h, 83
- pThis
  - aDCLoad.cpp, 72
- PULSE\_MAXIMUM
  - aDCLoad.h, 83
- pingAutolock
  - aDCSettings, 48
- pingBacklight
  - aDCDisplay, 29
- pingOperationMode
  - aDCSettings, 48
- Print.cpp, 85
- printCenter
  - aLCD, 60
- RXBUFFER\_MAXLEN
  - aDCEngine, 36
- reset
  - aStepper, 63
- restoreCalibration
  - aDCSettings, 49
- run
  - aDCEngine, 35
- SELECTION\_MODE\_CURRENT

- aDCSettings, [41](#)
- SELECTION\_MODE\_POWER
  - aDCSettings, [41](#)
- SELECTION\_MODE\_PULSE
  - aDCSettings, [41](#)
- SELECTION\_MODE\_UNKNOWN
  - aDCSettings, [41](#)
- SETTING\_ERROR\_OVERSIZED
  - aDCSettings, [41](#)
- SETTING\_ERROR\_UNDERSIZED
  - aDCSettings, [41](#)
- SETTING\_ERROR\_VALID
  - aDCSettings, [41](#)
- SelectionMode\_t
  - aDCSettings, [41](#)
- serialAvailable
  - aDCLoad.cpp, [69](#)
- serialFlush
  - aDCLoad.cpp, [69](#)
- serialPrint
  - aDCLoad.cpp, [69](#), [70](#)
- serialPrintln
  - aDCLoad.cpp, [70](#), [71](#)
- serialRead
  - aDCLoad.cpp, [71](#)
- serialWrite
  - aDCLoad.cpp, [71](#)
- service
  - aInputRelay, [58](#)
- setCalibrationMode
  - aDCSettings, [49](#)
- setCalibrationValues
  - aDCSettings, [49](#)
- setCurrent
  - aDCSettings, [49](#)
- setCurrentDAC
  - aDCSettings, [49](#)
- setCursor
  - aLCD, [60](#)
- setDisplayMode
  - aDCSettings, [50](#)
- setEncoderPosition
  - aDCSettings, [50](#)
- setFanSpeed
  - aDCSettings, [50](#)
- setInput
  - aDCEngine, [35](#)
  - aInputRelay, [58](#)
- setOperationMode
  - aDCSettings, [50](#)
- setPower
  - aDCSettings, [50](#)
- setPulse
  - aDCSettings, [52](#)
- setPulseHigh
  - aDCSettings, [52](#)
- setSelectionMode
  - aDCSettings, [52](#)
- setTemperature
  - aDCSettings, [52](#)
- setVoltage
  - aDCSettings, [53](#)
- SettingError\_t
  - aDCSettings, [41](#)
- setup
  - aDCDisplay, [30](#)
  - aDCEngine, [35](#)
  - sketch.cpp, [85](#)
- showBanner
  - aDCDisplay, [30](#)
- sketch.cpp, [85](#)
  - engine, [85](#)
  - loop, [85](#)
  - setup, [85](#)
- slope
  - aDCSettings::CalibrationData\_t, [64](#)
- Stream.cpp, [85](#)
- sync
  - aStepper, [63](#)
- syncData
  - aDCSettings, [53](#)
- TEMPERATURE\_MAXIMUM
  - aDCLoad.h, [84](#)
- timer1ISR
  - aDCLoad.cpp, [71](#)
- timer3ISR
  - aDCLoad.cpp, [71](#)
- Tone.cpp, [86](#)
- USB\_ICON\_X\_COORD
  - aDCLoad.h, [84](#)
- USB\_ICON\_Y\_COORD
  - aDCLoad.h, [84](#)
- USBCore.cpp, [86](#)
- updateDisplay
  - aDCDisplay, [30](#)
- updateField
  - aDCDisplay, [30](#)
- updateOperationMode
  - aDCSettings, [53](#)
- updateValuesFromMode
  - aDCSettings, [53](#)
- v
  - aDCSettings::\_eepromCalibrationValue\_t, [27](#)
- VOLTAGE\_MAXIMUM
  - aDCLoad.h, [84](#)
- WInterrupts.c, [86](#)
- WMath.cpp, [86](#)
- WString.cpp, [86](#)
- wiring.c, [86](#)
- wiring\_analog.c, [86](#)
- wiring\_digital.c, [86](#)
- wiring\_pulse.c, [86](#)
- wiring\_shift.c, [86](#)