

Manual for aWXIron

1.0

Written by F1RMB, Daniel Caujolle-Bert ©2015

Contents

1	SMD Soldering Station for Weller RT Series Tips	1
2	User Interface overview	3
2.1	Encoder usage	3
2.2	Joined mode	4
2.3	Standby	4
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Class Documentation	9
5.1	aDSChannel Class Reference	9
5.1.1	Detailed Description	11
5.1.2	Member Enumeration Documentation	11
5.1.2.1	HeatingState_t	11
5.1.3	Constructor & Destructor Documentation	11
5.1.3.1	aDSChannel	11
5.1.3.2	~aDSChannel	11
5.1.4	Member Function Documentation	12
5.1.4.1	_analogRead	12
5.1.4.2	_analogWrite	12
5.1.4.3	_digitalRead	12
5.1.4.4	_digitalWrite	12
5.1.4.5	_turnOffPWM	13
5.1.4.6	getADCValue	13
5.1.4.7	getCalibration	13
5.1.4.8	getHeatState	13
5.1.4.9	getLEDState	13
5.1.4.10	getStandbyMode	13
5.1.4.11	getTemperature	13

5.1.4.12	hasFocus	14
5.1.4.13	isTempHasChanged	14
5.1.4.14	service	14
5.1.4.15	setCalibration	14
5.1.4.16	setFocus	14
5.1.4.17	setStandbyMode	14
5.1.4.18	setTemperature	15
5.1.4.19	setup	15
5.1.4.20	syncTempChange	15
5.1.4.21	updateLEDState	15
5.1.5	Member Data Documentation	16
5.1.5.1	ADC_TO_TEMP_GAIN	16
5.1.5.2	ADC_TO_TEMP_OFFSET	16
5.1.5.3	BLINK_UPDATE_RATE	16
5.1.5.4	CNTRL_GAIN	16
5.1.5.5	DEFAULT_TEMPERATURE_OFFSET	16
5.1.5.6	DEFAULT_TEMPERATURE_SLOPE	16
5.1.5.7	m_adcValue	16
5.1.5.8	m_blinkStandby	16
5.1.5.9	m_cal	16
5.1.5.10	m_channel	16
5.1.5.11	m_currentTemp	16
5.1.5.12	m_hasFocus	16
5.1.5.13	m_heatState	16
5.1.5.14	m_inStandby	16
5.1.5.15	m_ledPin	16
5.1.5.16	m_ledState	16
5.1.5.17	m_nextBlink	16
5.1.5.18	m_nextLowering	16
5.1.5.19	m_nextPass	16
5.1.5.20	m_nextTempStep	16
5.1.5.21	m_pwmPin	16
5.1.5.22	m_pwmValue	16
5.1.5.23	m_ref	17
5.1.5.24	m_sensorPin	17
5.1.5.25	m_targetTemp	17
5.1.5.26	m_tempHasChanged	17
5.1.5.27	PWM_MAX_VALUE	17
5.1.5.28	TEMPERATURE_MAX	17
5.1.5.29	TEMPERATURE_MIN	17

5.1.5.30	TEMPERATURE_STANDBY	17
5.1.5.31	TEMPERATURE_TOLERANCE	17
5.2	aDSChannels Class Reference	17
5.2.1	Member Enumeration Documentation	21
5.2.1.1	Channel_t	21
5.2.2	Constructor & Destructor Documentation	21
5.2.2.1	aDSChannels	21
5.2.2.2	~aDSChannels	21
5.2.3	Member Function Documentation	21
5.2.3.1	_backupCalibrationFromEEPROM	21
5.2.3.2	_checkForMagicNumbers	22
5.2.3.3	_clearValue	22
5.2.3.4	_crc8	22
5.2.3.5	_displayBigDigit	22
5.2.3.6	_displayBigDigits	23
5.2.3.7	_enableData	23
5.2.3.8	_enableDataCheck	23
5.2.3.9	_getTempFromEEPROM	23
5.2.3.10	_read	24
5.2.3.11	_restoreCalibrationFromEEPROM	24
5.2.3.12	_scissor	24
5.2.3.13	_setTempToEEPROM	24
5.2.3.14	_showBanner	25
5.2.3.15	_updateDisplay	25
5.2.3.16	_updateField	25
5.2.3.17	_wakeupFromStandby	25
5.2.3.18	_write	25
5.2.3.19	_writeMagicNumbers	26
5.2.3.20	getCalibrationValues	26
5.2.3.21	getOperationMode	26
5.2.3.22	incEncoderPosition	26
5.2.3.23	isInCalibration	26
5.2.3.24	isInStandby	27
5.2.3.25	isJoined	27
5.2.3.26	pingOperationMode	27
5.2.3.27	restoreCalibrationValues	27
5.2.3.28	saveCalibrationValues	27
5.2.3.29	service	27
5.2.3.30	setCalibrationMode	27
5.2.3.31	setCalibrationValues	28

5.2.3.32	setFocusToNextChannel	28
5.2.3.33	setOperationMode	28
5.2.3.34	setup	28
5.2.3.35	syncData	29
5.2.3.36	toggleJoined	29
5.2.3.37	toggleStandbyMode	29
5.2.3.38	updateOperationMode	29
5.2.4	Member Data Documentation	29
5.2.4.1	DATA_CHANNEL1_LED_STATE	29
5.2.4.2	DATA_CHANNEL1_TEMP_READ	30
5.2.4.3	DATA_CHANNEL1_TEMP_SET	30
5.2.4.4	DATA_CHANNEL2_ENABLED	30
5.2.4.5	DATA_CHANNEL2_LED_STATE	30
5.2.4.6	DATA_CHANNEL2_TEMP_READ	30
5.2.4.7	DATA_CHANNEL2_TEMP_SET	30
5.2.4.8	DATA_CHANNELS_JOINED	30
5.2.4.9	DATA_DISPLAY	30
5.2.4.10	DATA_DISPLAY_STANDBY	30
5.2.4.11	DATA_FOCUS	30
5.2.4.12	DATA_IN_CALIBRATION	30
5.2.4.13	DATA_OPERATION	30
5.2.4.14	DATA_STANDBY	31
5.2.4.15	DISPLAY_UPDATE_RATE	31
5.2.4.16	EEPROM_ADDR_CALIBRATION_CHAN_1	31
5.2.4.17	EEPROM_ADDR_CALIBRATION_CHAN_2	31
5.2.4.18	EEPROM_ADDR_CHANNEL_JOINED	31
5.2.4.19	EEPROM_ADDR_MAGIC	31
5.2.4.20	EEPROM_ADDR_TEMP_CHANNEL_ONE	31
5.2.4.21	EEPROM_ADDR_TEMP_CHANNEL_TWO	31
5.2.4.22	EEPROM_CALIBRATION_SIZE	31
5.2.4.23	EEPROM_STORAGE_STARTING	31
5.2.4.24	EEPROM_TEMP_SIZE	31
5.2.4.25	m_channels	32
5.2.4.26	m_datas	32
5.2.4.27	m_isValidEEPROM	32
5.2.4.28	m_lastTempChange	32
5.2.4.29	m_lcd	32
5.2.4.30	m_lcdCols	32
5.2.4.31	m_lcdRows	32
5.2.4.32	m_nextDisplayUpdate	32

5.2.4.33	<code>m_nextMeasureUpdate</code>	32
5.2.4.34	<code>m_operationMode</code>	32
5.2.4.35	<code>m_operationTick</code>	32
5.2.4.36	<code>m_storedToEEPROM</code>	32
5.2.4.37	<code>MEASURE_UPDATE_RATE</code>	32
5.2.4.38	<code>OFFSET_MARKER_LEFT</code>	32
5.2.4.39	<code>OFFSET_MARKER_RIGHT</code>	32
5.2.4.40	<code>OFFSET_VALUE</code>	32
5.2.4.41	<code>OPERATION_SET_TIMEOUT</code>	32
5.2.4.42	<code>TEMP_SETTING_INACTIVITY</code>	32
5.3	<code>aDSEngine</code> Class Reference	33
5.3.1	Constructor & Destructor Documentation	33
5.3.1.1	<code>aDSEngine</code>	33
5.3.1.2	<code>~aDSEngine</code>	33
5.3.2	Member Function Documentation	34
5.3.2.1	<code>_handleSerialInput</code>	34
5.3.2.2	<code>run</code>	34
5.3.2.3	<code>setup</code>	34
5.3.3	Member Data Documentation	34
5.3.3.1	<code>m_channels</code>	34
5.3.3.2	<code>m_datas</code>	34
5.3.3.3	<code>m_encoder</code>	34
5.3.3.4	<code>m_RXbuffer</code>	34
5.3.3.5	<code>m_RXoffset</code>	34
5.3.3.6	<code>m_serialInputTick</code>	34
5.3.3.7	<code>RXBUFFER_MAXLEN</code>	34
5.4	<code>aDSChannel::aPin_t</code> Struct Reference	35
5.4.1	Detailed Description	35
5.4.2	Member Data Documentation	35
5.4.2.1	<code>mask</code>	35
5.4.2.2	<code>outputRegister</code>	35
5.4.2.3	<code>pin</code>	35
5.4.2.4	<code>port</code>	35
5.4.2.5	<code>timer</code>	35
6	File Documentation	37
6.1	<code>aDSEngine.cpp</code> File Reference	37
6.1.1	Detailed Description	37
6.1.2	Function Documentation	38
6.1.2.1	<code>getNumericalLength</code>	38

6.1.2.2	timer1ISR	38
6.1.3	Variable Documentation	38
6.1.3.1	_bigDigitsBottom	38
6.1.3.2	_bigDigitsTop	39
6.1.3.3	DIGIT_WIDTH	39
6.1.3.4	pEncoder	39
6.1.3.5	PROGMEM	39
6.2	aDSEngine.h File Reference	39
6.2.1	Detailed Description	41
6.2.2	Macro Definition Documentation	41
6.2.2.1	IS_DATA_ENABLED	41
6.2.2.2	SIMU	41
6.2.3	Enumeration Type Documentation	41
6.2.3.1	OperationMode_t	41
6.2.4	Variable Documentation	41
6.2.4.1	CHANNEL2_ENABLE_PIN	41
6.2.4.2	ENCODER_A_PIN	42
6.2.4.3	ENCODER_B_PIN	42
6.2.4.4	ENCODER_PB_PIN	42
6.2.4.5	ENCODER_STEPS_PER_NOTCH	42
6.2.4.6	LCD_COLS	42
6.2.4.7	LCD_D4_PIN	42
6.2.4.8	LCD_D5_PIN	42
6.2.4.9	LCD_D6_PIN	42
6.2.4.10	LCD_D7_PIN	42
6.2.4.11	LCD_ENABLE_PIN	42
6.2.4.12	LCD_ROWS	42
6.2.4.13	LCD_RS_PIN	42
6.2.4.14	LED_CHANNEL1_PIN	43
6.2.4.15	LED_CHANNEL2_PIN	43
6.2.4.16	PROGRAM_VERSION_MAJOR	43
6.2.4.17	PROGRAM_VERSION_MINOR	43
6.2.4.18	PWM_CHANNEL1_PIN	43
6.2.4.19	PWM_CHANNEL2_PIN	43
6.2.4.20	TEMP_SENSOR_CHANNEL1_PIN	43
6.2.4.21	TEMP_SENSOR_CHANNEL2_PIN	43
6.3	CDC.cpp File Reference	43
6.4	HardwareSerial.cpp File Reference	43
6.5	HID.cpp File Reference	43
6.6	IPAddress.cpp File Reference	44

6.7	libraries.cpp File Reference	44
6.8	main.cpp File Reference	44
6.9	new.cpp File Reference	44
6.10	Print.cpp File Reference	44
6.11	sketch.cpp File Reference	44
6.11.1	Function Documentation	44
6.11.1.1	loop	44
6.11.1.2	setup	45
6.11.2	Variable Documentation	45
6.11.2.1	engine	45
6.12	Stream.cpp File Reference	45
6.13	Tone.cpp File Reference	45
6.14	USBCore.cpp File Reference	45
6.15	WInterrupts.c File Reference	45
6.16	wiring.c File Reference	45
6.16.1	Macro Definition Documentation	46
6.16.1.1	FRACT_INC	46
6.16.1.2	FRACT_MAX	46
6.16.1.3	MICROSECONDS_PER_TIMER0_OVERFLOW	46
6.16.1.4	MILLIS_INC	46
6.16.2	Function Documentation	46
6.16.2.1	delay	46
6.16.2.2	delayMicroseconds	46
6.16.2.3	if	46
6.16.2.4	init	46
6.16.2.5	micros	46
6.16.2.6	millis	46
6.16.3	Variable Documentation	46
6.16.3.1	f	46
6.16.3.2	m	46
6.16.3.3	timer0_fract	46
6.16.3.4	timer0_millis	46
6.16.3.5	timer0_overflow_count	46
6.17	wiring_analog.c File Reference	46
6.18	wiring_digital.c File Reference	46
6.19	wiring_pulse.c File Reference	47
6.20	wiring_shift.c File Reference	47
6.21	WMath.cpp File Reference	47
6.22	WString.cpp File Reference	47

[Index](#)

48

Chapter 1

SMD Soldering Station for Weller RT Series Tips

Based on a project from **Martin Kumm** http://www.martin-kumm.de/wiki/doku.php?id=Projects:SMD_Solderstation.

The hardware has been redesigned (two channels, 16x2 LCD instead of 7 segments display, etc), and the software rewrote from scratch.

I want to especially thank my friend **Olivier, F5LGJ**, for his great help and support in this project.

Chapter 2

User Interface overview

- The Soldering Station control is done using a simple rotary encoder, which integrates a push button.
- The temperature range goes from 100 °C, up to 450 °C.
- Depending of the hardware assembly, it can control one or two soldering irons:
 1. In Single mode, the temperature reading and setting is displayed using double height font.
 2. In dual channels version, both soldering irons can be controled separately, or can be joined:
 - when separate channels mode is used, each channel is independent. Simple click on the encoder push button will set the focus to the next channel. The focused channel temperature will be surrounded by the symbols [and]
 - when joined mode is used, the temperature is displayed like in Single channel mode (double height font), both channels share the same settings (target temperature and standby mode).
- LED status decoding:

LED Status	Meaning
ON	the tip is heating
OFF	the tip is cooling
Blinking	the tip has reached his target temperature
Three times blinking	the soldering station is in Standby mode (see Standby)

- The target temperature is stored, for each channel, inside the microcontroller's EEPROM. The values will be restored on the next startup. After a timeout of 60 seconds, a new defined target temperature will be stored into the EEPROM. If in the meantime the user defines a new target temperature, the timeout is resetted
- When the station is in temperature reading mode, the displayed value(s) is left aligned. When the station is in settings mode, the displayed value is right aligned.

2.1 Encoder usage

- In any mode (settings or temperature reading), the rotary encoder is used to define the target temperature. Turn the encoder clockwise to increase the target temperature, and anti-clockwise to decrease it.
- When the soldering station is not in settings mode, it displays the soldering tip temperature. A single encoder detents rotation will switch the soldering station into settings mode, and display the target temperature without any change to the target temperature setting.

- When the soldering station is in settings mode, if not action is done using the encoder's rotation within 3 seconds, it will switch back to temperature reading.

- Encoder's push button:

- Single soldering tip version:

Button	Action
Single Click	<i>no effect</i>
Double Click	toggles standby mode (see Standby)
Held	<i>no effect</i>

- Dual soldering tip version:

Button	Action
Single click	change the focus to the next channel (if not in joined mode)
Double click	switch to standby mode (see Standby)
Held	toggles joined mode (see Joined mode)

2.2 Joined mode

- With dual channel enabled hardware, it's possible to share the same temperature preset for both soldering tips.

See [Encoder usage](#)

2.3 Standby

- A double-click on the encoder brings the soldering station in the standby mode.
- When standby mode is enabled, the target temperature will go down to 150 °C if the temperature setting is set above this point, otherwise it will go down to 100 °C.
- Any encoder action will exit the standby mode.
- When Standby mode is activated, the LEDs blink three times cyclically.

Chapter 3

Class Index

3.1 Class List

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

aDSChannels::_eepromCalibrationValue_t	
Union to manipulate float/uint8_t [] calibration values	??
aDSChannel	
ADSCChannel class	9
aDSChannels	17
aDSEngine	33
aDSChannel::aPin_t	
Our pin structure	35
aDSChannel::CalibrationData_t	
Calibration values	??

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

aDSEngine.cpp	37
aDSEngine.h	39
CDC.cpp	43
HardwareSerial.cpp	43
HID.cpp	43
IPAddress.cpp	44
libraries.cpp	44
main.cpp	44
new.cpp	44
Print.cpp	44
sketch.cpp	44
Stream.cpp	45
Tone.cpp	45
USBCore.cpp	45
WInterrupts.c	45
wiring.c	45
wiring_analog.c	46
wiring_digital.c	46
wiring_pulse.c	47
wiring_shift.c	47
WMath.cpp	47
WString.cpp	47

Chapter 5

Class Documentation

5.1 aDSChannel Class Reference

[aDSChannel](#) class

```
#include <aDSEngine.h>
```

Classes

- struct [aPin_t](#)
Our pin structure.
- struct [CalibrationData_t](#)
Calibration values.

Public Types

- enum [HeatingState_t](#) { [HEATING_STATE_HEATING](#), [HEATING_STATE_COOLING](#), [HEATING_STATE_REACHED](#), [HEATING_STATE_STANDBY](#) }
Heating State enumeration.

Public Member Functions

- [aDSChannel](#) ()
aDSChannel class constructor
- virtual [~aDSChannel](#) ()
aDSChannel destructor
- void [setup](#) (uint8_t, uint8_t, uint8_t)
Setup member function, should be called before any other member.
- void [setFocus](#) (bool)
Set the focus, as display point of view.
- bool [hasFocus](#) ()
Get the focus state.
- uint16_t [getTemperature](#) ([OperationMode_t](#))
Get current temperature accordingly from the given mode (SET/READ)
- bool [setTemperature](#) ([OperationMode_t](#), int16_t)
Set current temperature accordingly from the given mode (SET/READ)
- bool [service](#) (unsigned long)

This member should be called often, it manage heating/cooling of the Channel.

- void `setStandbyMode` (bool)
Enable or disable channel's standby.
- bool `getStandbyMode` ()
Get channel's standby enableity.
- bool `isTempHasChanged` ()
Is target temperature has changed (use for EEPROM storage)
- void `syncTempChange` ()
Reset temperature change flag (use for EEPROM storage)
- uint8_t `updateLEDState` (unsigned long)
Change LED state according for Heating/Cooling/Standby status.
- uint8_t `getLEDState` ()
Get state LED status.
- `HeatingState_t` `getHeatState` ()
Get channel heating state.
- void `setCalibration` (float, float)
- const `CalibrationData_t` `getCalibration` () const
Get calibration data values.
- int16_t `getADCValue` ()
Get latest ADC value.

Static Public Attributes

- static const int16_t `TEMPERATURE_MIN` = 10
- static const int16_t `TEMPERATURE_MAX` = 450
Maximum temperature.
- static const int16_t `TEMPERATURE_STANDBY` = 150
Standby temperature.
- static const float `ADC_TO_TEMP_GAIN` = 0.39
- static const float `ADC_TO_TEMP_OFFSET` = 23.9
- static const uint8_t `CNTRL_GAIN` = 10
- static const unsigned long `BLINK_UPDATE_RATE` = 400
Update rate for LED blinking, in ms.
- static const int16_t `TEMPERATURE_TOLERANCE` = 3
Temperature tolerance for REACHED state, +/- 2 °C.
- static const float `DEFAULT_TEMPERATURE_SLOPE` = 0.3947387545
- static const float `DEFAULT_TEMPERATURE_OFFSET` = 43.8279285472
- static const uint8_t `PWM_MAX_VALUE` = 150
Maximum PWM value.

Protected Member Functions

- void `_turnOffPWM` (aPin_t)
Turns PWM off for the given pin.
- int8_t `_digitalRead` (aPin_t)
Get state of the given pin.
- void `_digitalWrite` (aPin_t, uint8_t)
Set state of the given pin.
- uint16_t `_analogRead` (aPin_t)
Get analog value of the given pin.
- void `_analogWrite` (aPin_t, uint8_t)
Set analog value for the given pin.

Private Attributes

- [aPin_t m_pwmPin](#)
- [aPin_t m_sensorPin](#)
- [aPin_t m_ledPin](#)
- [bool m_hasFocus](#)
- [int16_t m_targetTemp](#)
- [int16_t m_currentTemp](#)
- [int8_t m_pwmValue](#)
- [int16_t m_adcValue](#)
- [bool m_inStandby](#)
- [HeatingState_t m_heatState](#)
- [uint8_t m_ledState](#)
- [unsigned long m_nextPass](#)
- [bool m_tempHasChanged](#)
- [unsigned long m_nextBlink](#)
- [uint8_t m_blinkStandby](#)
- [uint8_t m_ref](#)
- [CalibrationData_t m_cal](#)
- [uint8_t m_channel](#)
- [unsigned long m_nextTempStep](#)
- [unsigned long m_nextLowering](#)

5.1.1 Detailed Description

[aDSChannel](#) class

5.1.2 Member Enumeration Documentation

5.1.2.1 enum aDSChannel::HeatingState_t

Heating State enumeration.

Enumerator

- HEATING_STATE_HEATING*** Heating.
- HEATING_STATE_COOLING*** Cooling.
- HEATING_STATE_REACHED*** Target temperature reached.
- HEATING_STATE_STANDBY*** In standby mode.

5.1.3 Constructor & Destructor Documentation

5.1.3.1 aDSChannel::aDSChannel ()

[aDSChannel](#) class constructor

5.1.3.2 aDSChannel::~~aDSChannel () [virtual]

[aDSChannel](#) destructor

5.1.4 Member Function Documentation

5.1.4.1 `uint16_t aDSChannel::_analogRead (aPin_t pin)` [protected]

Get analog value of the given pin.

Took and Hacked from Arduino [wiring_analog.c](#)

Parameters

<i>pin</i>	aPin_t : pin
------------	------------------------------

Returns

`uint8_t` : analog value (0..255)

5.1.4.2 `void aDSChannel::_analogWrite (aPin_t pin, uint8_t val)` [protected]

Set analog value for the given pin.

Took and Hacked from Arduino [wiring_analog.c](#)

Parameters

<i>pin</i>	aPin_t : pin
<i>val</i>	<code>uint8_t</code> : analog value (0..255)

Returns

`void`

5.1.4.3 `int8_t aDSChannel::_digitalRead (aPin_t pin)` [protected]

Get state of the given pin.

Took and Hacked from Arduino [wiring_digital.c](#)

Parameters

<i>pin</i>	aPin_t : pin
------------	------------------------------

Returns

`int8_t` : HIGH or LOW

5.1.4.4 `void aDSChannel::_digitalWrite (aPin_t pin, uint8_t val)` [protected]

Set state of the given pin.

Took and Hacked from Arduino [wiring_digital.c](#)

Parameters

<i>pin</i>	aPin_t : pin
<i>val</i>	<code>uint8_t</code> : state (HIGH or LOW)

Returns

`void`

5.1.4.5 void aDSChannel::_turnOffPWM (aPin_t *pin*) [protected]

Turns PWM off for the given pin.

Took and Hacked from Arduino [wiring_digital.c](#)

Parameters

<i>pin</i>	aPin_t : pin
------------	------------------------------

Returns

void

5.1.4.6 int16_t aDSChannel::getADCValue ()

Get latest ADC value.

Returns

int16_t : ADC value

5.1.4.7 const aDSChannel::CalibrationData_t aDSChannel::getCalibration () const

Get calibration data values.

Returns

const [CalibrationData_t](#) : Calibration data

5.1.4.8 aDSChannel::HeatingState_t aDSChannel::getHeatState ()

Get channel heating state.

Returns

[aDSChannel::HeatingState_t](#) : heating state

5.1.4.9 uint8_t aDSChannel::getLEDState ()

Get state LED status.

Returns

uint8_t : HIGH or LOW (LED on or off, accordingly)

5.1.4.10 bool aDSChannel::getStandbyMode ()

Get channel's standby enability.

Returns

bool

5.1.4.11 uint16_t aDSChannel::getTemperature (OperationMode_t *mode*)

Get current temperature accordingly from the given mode (SET/READ)

Parameters

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

Returns

uint16_t : temperature, in Celcius

5.1.4.12 bool aDSChannel::hasFocus ()

Get the focus state.

Returns

bool : focus state

5.1.4.13 bool aDSChannel::isTempHasChanged ()

Is target temperature has changed (use for EEPROM storage)

Returns

bool : return true if target temperature has changed, otherwise false

5.1.4.14 bool aDSChannel::service (unsigned long *m*)

This member should be called often, it manage heating/cooling of the Channel.

Parameters

<i>m</i>	unsigned long : current timestamp
----------	-----------------------------------

Returns

bool : return true if readed temperature has changed since last call, otherwise false

5.1.4.15 void aDSChannel::setCalibration (float *slope*, float *offset*)5.1.4.16 void aDSChannel::setFocus (bool *v*)

Set the focus, as display point of view.

Parameters

<i>v</i>	bool : focus state
----------	--------------------

Returns

void

5.1.4.17 void aDSChannel::setStandbyMode (bool *enable*)

Enable or disable channel's standby.

Parameters

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

Returns

void

5.1.4.18 bool aDSChannel::setTemperature (**OperationMode_t** *mode*, int16_t *temp*)

Set current temperature accordingly from the given mode (SET/READ)

Parameters

<i>mode</i>	OperationMode_t : operation mode
<i>temp</i>	int16_t : temperature, in Celcius

Returns

bool : true if temperature has been changed, otherwise false

5.1.4.19 void aDSChannel::setup (uint8_t *pwmPin*, uint8_t *sensorPin*, uint8_t *ledPin*)

Setup member function, should be called before any other member.

Pins will be embedded into [aPin_t](#) object, timer, mask, port and output register will be set also here, preventing using Arduino analog/digital{Read/Write}() calls, which are quite slow.

Parameters

<i>pwmPin</i>	uint8_t : PWM pin, used to drive the output MosFET
<i>sensorPin</i>	uint8_t : Sensor pin, used to get analog temperature value.
<i>ledPin</i>	uint8_t : LED pin, used to reflect Heating/Cooling state

Returns

void

5.1.4.20 void aDSChannel::syncTempChange ()

Reset temperature change flag (use for EEPROM storage)

Returns

void

5.1.4.21 uint8_t aDSChannel::updateLEDState (unsigned long *m*)

Change LED state according for Heating/Cooling/Standby status.

Parameters

<i>m</i>	unsigned long : timestamp
----------	---------------------------

Returns

uint8_t : HIGH or LOW (LED on or off, accordingly)

5.1.5 Member Data Documentation

5.1.5.1 `const float aDSChannel::ADC_TO_TEMP_GAIN = 0.39` [static]

5.1.5.2 `const float aDSChannel::ADC_TO_TEMP_OFFSET = 23.9` [static]

5.1.5.3 `const unsigned long aDSChannel::BLINK_UPDATE_RATE = 400` [static]

Update rate for LED blinking, in ms.

5.1.5.4 `const uint8_t aDSChannel::CNTRL_GAIN = 10` [static]

5.1.5.5 `const float aDSChannel::DEFAULT_TEMPERATURE_OFFSET = 43.8279285472` [static]

5.1.5.6 `const float aDSChannel::DEFAULT_TEMPERATURE_SLOPE = 0.3947387545` [static]

5.1.5.7 `int16_t aDSChannel::m_adcValue` [private]

5.1.5.8 `uint8_t aDSChannel::m_blinkStandby` [private]

5.1.5.9 `CalibrationData_t aDSChannel::m_cal` [private]

5.1.5.10 `uint8_t aDSChannel::m_channel` [private]

5.1.5.11 `int16_t aDSChannel::m_currentTemp` [private]

5.1.5.12 `bool aDSChannel::m_hasFocus` [private]

5.1.5.13 `HeatingState_t aDSChannel::m_heatState` [private]

5.1.5.14 `bool aDSChannel::m_inStandby` [private]

5.1.5.15 `aPin_t aDSChannel::m_ledPin` [private]

5.1.5.16 `uint8_t aDSChannel::m_ledState` [private]

5.1.5.17 `unsigned long aDSChannel::m_nextBlink` [private]

5.1.5.18 `unsigned long aDSChannel::m_nextLowering` [private]

5.1.5.19 `unsigned long aDSChannel::m_nextPass` [private]

5.1.5.20 `unsigned long aDSChannel::m_nextTempStep` [private]

5.1.5.21 `aPin_t aDSChannel::m_pwmPin` [private]

5.1.5.22 `int8_t aDSChannel::m_pwmValue` [private]

5.1.5.23 `uint8_t aDSChannel::m_ref` [private]

5.1.5.24 `aPin_t aDSChannel::m_sensorPin` [private]

5.1.5.25 `int16_t aDSChannel::m_targetTemp` [private]

5.1.5.26 `bool aDSChannel::m_tempHasChanged` [private]

5.1.5.27 `const uint8_t aDSChannel::PWM_MAX_VALUE = 150` [static]

Maximum PWM value.

5.1.5.28 `const int16_t aDSChannel::TEMPERATURE_MAX = 450` [static]

Maximum temperature.

5.1.5.29 `const int16_t aDSChannel::TEMPERATURE_MIN = 10` [static]

5.1.5.30 `const int16_t aDSChannel::TEMPERATURE_STANDBY = 150` [static]

Standby temperature.

5.1.5.31 `const int16_t aDSChannel::TEMPERATURE_TOLERANCE = 3` [static]

Temperature tolerance for REACHED state, +/- 2 °C.

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

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

5.2 aDSChannels Class Reference

```
#include <aDSEngine.h>
```

Classes

- [union _eepromCalibrationValue_t](#)
Union to manipulate float/uint8_t [] calibration values.

Public Types

- `enum Channel_t { CHANNEL_ONE, CHANNEL_TWO, CHANNEL_MAX }`
Channels enumeration.

Public Member Functions

- [aDSChannels](#) (uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t)
aDSChannels constructor
- `virtual ~aDSChannels ()`
aDSChannels destructor

- void [setup](#) (uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t, uint8_t)
Setup member function, should be called before any other member.
- void [setOperationMode](#) ([OperationMode_t](#))
Set operation mode (SET/READ)
- [OperationMode_t](#) [getOperationMode](#) ()
Get current operation mode.
- void [updateOperationMode](#) ()
Update operation mode.
- void [pingOperationMode](#) ()
Reset timeout for OPERATION_MODE_SET mode.
- void [syncData](#) (uint16_t)
Reset given bit inside bitfield, regardless of its state.
- void [incEncoderPosition](#) (uint16_t)
Increment or decrement encoder position.
- void [service](#) ()
This member should be called often, it manage channels, LCD and EEPROM.
- void [toggleJoined](#) ()
Toggle channels joining.
- bool [isJoined](#) ()
Get channel joining state.
- void [setFocusToNextChannel](#) ()
Set the focus to the next channel, if any.
- void [toggleStandbyMode](#) ()
Toggle standby mode.
- bool [isInStandby](#) ()
Get standby mode.
- void [setCalibrationValues](#) ([Channel_t](#), [aDSChannel::CalibrationData_t](#))
Set calibration values for given channel.
- [aDSChannel::CalibrationData_t](#) [getCalibrationValues](#) ([Channel_t](#))
Get calibration value for given channel.
- void [restoreCalibrationValues](#) ()
Restore calibration values from EEPROM.
- void [saveCalibrationValues](#) ([Channel_t](#))
Save calibration value of given into EEPROM.
- bool [isInCalibration](#) ()
Get calibration mode enableity.
- void [setCalibrationMode](#) (bool)
Set calibration mode.

Static Public Attributes

- static const uint8_t [OFFSET_VALUE](#) = 2
Value column LCD offset.
- static const uint8_t [OFFSET_MARKER_LEFT](#) = 0
Column LCD offset for left marker '['.
- static const uint8_t [OFFSET_MARKER_RIGHT](#) = 10
Column LCD offset for right marker ']'.
- static const unsigned long [OPERATION_SET_TIMEOUT](#) = 3000
Automatic toggle settings->reading timeout (3 seconds), in ms.
- static const unsigned long [DISPLAY_UPDATE_RATE](#) = 200

- *Display update rate, in ms.*
static const unsigned long `MEASURE_UPDATE_RATE` = 200
- *Measurement (for `aDSChannel`) rate, in ms.*
static const unsigned long `TEMP_SETTING_INACTIVITY` = 60000
- *Timeout in ms, after which the new target temperature will be stored in the EEPROM.*
static const uint16_t `DATA_CHANNEL2_ENABLED` = 1
Bitfield: Channel 2 is enabled.
- static const uint16_t `DATA_CHANNELS_JOINED` = 1 << 1
Bitfield: Channel 1 & 2 are joined.
- static const uint16_t `DATA_OPERATION` = 1 << 2
Bitfield: Operation mode has changed.
- static const uint16_t `DATA_CHANNEL1_TEMP_SET` = 1 << 3
Bitfield: Target temperature of channel 1 has changed.
- static const uint16_t `DATA_CHANNEL1_TEMP_READ` = 1 << 4
Bitfield: Readed temperature of channel 1 has changed.
- static const uint16_t `DATA_CHANNEL1_LED_STATE` = 1 << 5
Bitfield: LED state of channel 1 has changed.
- static const uint16_t `DATA_CHANNEL2_TEMP_SET` = 1 << 6
Bitfield: Target temperature of channel 1 has changed.
- static const uint16_t `DATA_CHANNEL2_TEMP_READ` = 1 << 7
Bitfield: Readed temperature of channel 1 has changed.
- static const uint16_t `DATA_CHANNEL2_LED_STATE` = 1 << 8
Bitfield: LED state of channel 1 has changed.
- static const uint16_t `DATA_DISPLAY` = 1 << 9
Bitfield: Display should be refreshed.
- static const uint16_t `DATA_STANDBY` = 1 << 10
Bitfield: Standby state.
- static const uint16_t `DATA_DISPLAY_STANDBY` = 1 << 11
Bitfield: Standby state has changed.
- static const uint16_t `DATA_FOCUS` = 1 << 12
Bitfield: Focus has changed.
- static const uint16_t `DATA_IN_CALIBRATION` = 1 << 13
Bitfield: in Calibration.

Protected Member Functions

- void `_enableData` (uint16_t, bool)
Enable a bit, regardless of its state, inside bitfield `m_datas`.
- void `_enableDataCheck` (uint16_t, bool)
Enable a bit, if it's not already set, inside bitfield `m_datas`.
- void `_updateDisplay` ()
Update LCD display, if needed.
- void `_displayBigDigit` (uint8_t, uint8_t, uint8_t=0)
Display a big digit to given position.
- void `_displayBigDigits` (uint16_t, uint8_t, uint8_t=0)
Display a big digits number to given position.
- void `_clearValue` (uint8_t, int=0)
Clear numerical value field (in non big digit mode) on LCD.
- void `_updateField` (OperationMode_t, int16_t, uint8_t)
Update value on LCD from given mode and row.

- void `_wakeupFromStandby ()`
Wake up from standby mode.
- void `_showBanner ()`
Display a banner on the LCD.
- bool `_checkForMagicNumbers ()`
Check for the magic number in the EEPROM.
- void `_writeMagicNumbers ()`
Write magic numbers into EEPROM.
- uint8_t `_crc8 (const uint8_t *, uint8_t)`
CRC8 computation.
- template<typename T >
bool `_write (T const, int16_t &)`
Template to write a value into EEPROM, at given address.
- template<typename T >
bool `_read (T &, int16_t &)`
Template to read a value from the EEPROM, at given address.
- template<typename T >
void `_scissor (T v, uint8_t *, size_t &)`
Template to decompose the value into an array of uint8_t (used for CRC8 computation)
- bool `_getTempFromEEPROM (int16_t, uint16_t &)`
Helper to read the stored temperature inside EEPROM at given address.
- void `_setTempToEEPROM (int16_t, uint16_t)`
Helper to write a temperature inside EEPROM at given address.
- void `_restoreCalibrationFromEEPROM (int16_t, aDSChannel &)`
Restore calibration value for given channel.
- void `_backupCalibrationFromEEPROM (int16_t, aDSChannel &)`
Backup calibration value for given channel.

Static Protected Attributes

- static const int16_t `EEPROM_ADDR_MAGIC = 0`
EEPROM offset storage start for magic numbers (0xDEAD)
- static const int16_t `EEPROM_STORAGE_STARTING = 5`
EEPROM starting address for program datas.
- static const int16_t `EEPROM_TEMP_SIZE = sizeof(uint16_t) + sizeof(uint8_t)`
EEPROM temperature size (temperature + crc)
- static const int16_t `EEPROM_ADDR_CHANNEL_JOINED = EEPROM_STORAGE_STARTING + 1`
Channels are joined.
- static const int16_t `EEPROM_ADDR_TEMP_CHANNEL_ONE = EEPROM_ADDR_CHANNEL_JOINED + EEPROM_TEMP_SIZE`
Target temp for Channel 1.
- static const int16_t `EEPROM_ADDR_TEMP_CHANNEL_TWO = EEPROM_ADDR_TEMP_CHANNEL_ONE + EEPROM_TEMP_SIZE`
Target temp for Channel 2.
- 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_CHAN_1 = EEPROM_ADDR_TEMP_CHANNEL_TWO + EEPROM_TEMP_SIZE`
EEPROM start offset for Channel 1 calibration values.
- static const int16_t `EEPROM_ADDR_CALIBRATION_CHAN_2 = EEPROM_ADDR_CALIBRATION_CHAN_1 + EEPROM_CALIBRATION_SIZE`
EEPROM start offset for Channel 2 calibration values.

Private Attributes

- LiquidCrystal [m_lcd](#)
- [aDSChannel](#) [m_channels](#) [[CHANNEL_MAX](#)]
- [OperationMode_t](#) [m_operationMode](#)
- unsigned long [m_operationTick](#)
- [uint16_t](#) [m_datas](#)
- [uint8_t](#) [m_lcdCols](#)
- [uint8_t](#) [m_lcdRows](#)
- unsigned long [m_nextDisplayUpdate](#)
- unsigned long [m_nextMeasureUpdate](#)
- unsigned long [m_lastTempChange](#)
- bool [m_isValidEEPROM](#)
- bool [m_storedToEEPROM](#)

5.2.1 Member Enumeration Documentation

5.2.1.1 enum [aDSChannels::Channel_t](#)

Channels enumeration.

Enumerator

CHANNEL_ONE

CHANNEL_TWO

CHANNEL_MAX

5.2.2 Constructor & Destructor Documentation

5.2.2.1 [aDSChannels::aDSChannels](#) ([uint8_t rs](#), [uint8_t e](#), [uint8_t d4](#), [uint8_t d5](#), [uint8_t d6](#), [uint8_t d7](#))

[aDSChannels](#) constructor

Parameters

<i>rs</i>	uint8_t : LCD RS pin
<i>e</i>	uint8_t : LCD Enable pin
<i>d4</i>	uint8_t : LCD D4 pin
<i>d5</i>	uint8_t : LCD D5 pin
<i>d6</i>	uint8_t : LCD D6 pin
<i>d7</i>	uint8_t : LCD D7 pin

5.2.2.2 [aDSChannels::~~aDSChannels](#) () [[virtual](#)]

[aDSChannels](#) destructor

5.2.3 Member Function Documentation

5.2.3.1 void [aDSChannels::backupCalibrationFromEEPROM](#) ([int16_t startAddr](#), [aDSChannel & channel](#)) [[protected](#)]

Backup calibration value for given channel.

Parameters

<i>startAddr</i>	int16_t : EEPROM start address
<i>channel</i>	aDSChannel& : channel

Returns

void

5.2.3.2 bool aDSChannels::_checkForMagicNumbers () [protected]

Check for the magic number in the EEPROM.

Returns

bool : true if magic numbers has been found, otherwise false

5.2.3.3 void aDSChannels::_clearValue (uint8_t row, int destMinus = 0) [protected]

Clear numerical value field (in non big digit mode) on LCD.

Parameters

<i>row</i>	uint8_t : LCD row position
<i>destMinus</i>	int : right offset sub

Returns

void

5.2.3.4 uint8_t aDSChannels::_crc8 (const uint8_t* addr, uint8_t len) [protected]

CRC8 computation.

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

Parameters

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

Returns

uint8_t : **CRC**

5.2.3.5 void aDSChannels::_displayBigDigit (uint8_t digit, uint8_t position, uint8_t offset = 0) [protected]

Display a big digit to given position.

Parameters

<i>digit</i>	uint8_t : offset in _bigDigit{Top/Bottom} array
--------------	---

<i>position</i>	uint8_t : LCD position
<i>offset</i>	uint8_t : LCD offset position

Returns

void

5.2.3.6 void aDSChannels::_displayBigDigits (uint16_t *value*, uint8_t *position*, uint8_t *offset* = 0) [protected]

Display a big digits number to given position.

Parameters

<i>value</i>	uint16_t : value to display, DIGIT_WIDTH max length
<i>position</i>	uint8_t : LCD position
<i>offset</i>	uint8_t : LCD offset position

Returns

void

5.2.3.7 void aDSChannels::_enableData (uint16_t *bit*, bool *enable*) [protected]

Enable a bit, regardless of its state, inside bitfield m_datas.

Parameters

<i>bit</i>	uint16_t : bit to enable/disable
<i>enable</i>	bool : bit enableity

Returns

void

5.2.3.8 void aDSChannels::_enableDataCheck (uint16_t *bit*, bool *enable*) [protected]

Enable a bit, if it's not already set, inside bitfield m_datas.

Parameters

<i>bit</i>	uint16_t : bit to enable/disable
<i>enable</i>	bool : bit enableity

Returns

void

5.2.3.9 bool aDSChannels::_getTempFromEEPROM (int16_t *startAddr*, uint16_t & *temp*) [protected]

Helper to read the stored temperature inside EEPROM at given address.

Parameters

<i>startAddr</i>	int16_t : start address
<i>temp</i>	uint16_t& : temperature

Returns

bool : true if the CRCs matches

5.2.3.10 `template<typename T> bool aDSChannels::_read (T & v, int16_t & addr)` [protected]

Template to read a value from the EEPROM, at given address.

Parameters

<i>v</i>	T& : readed value
<i>addr</i>	int16_t& : start address

Returns

template <typename T> bool : true on read success, otherwise false

5.2.3.11 `void aDSChannels::_restoreCalibrationFromEEPROM (int16_t startAddr, aDSChannel & channel)`
[protected]

Restore calibration value for given channel.

Parameters

<i>startAddr</i>	int16_t : EEPROM start address
<i>channel</i>	aDSChannel& : channel

Returns

void

5.2.3.12 `template<typename T> void aDSChannels::_scissor (T v, uint8_t* dest, size_t & offset)` [protected]

Template to decompose the value into an array of uint8_t (used for CRC8 computation)

Parameters

<i>v</i>	T : value
<i>dest</i>	uint8_t* : pointer to the destination array
<i>offset</i>	size_t& : start offset of the array

Returns

template <typename T> void

5.2.3.13 `void aDSChannels::_setTempToEEPROM (int16_t startAddr, uint16_t temp)` [protected]

Helper to write a temperature inside EEPROM at given address.

Parameters

<i>startAddr</i>	int16_t : start address
<i>temp</i>	uint16_t : temperature

Returns

void

5.2.3.14 void aDSChannels::_showBanner () [protected]

Display a banner on the LCD.

Returns

void

5.2.3.15 void aDSChannels::_updateDisplay () [protected]

Update LCD display, if needed.

Returns

void

5.2.3.16 void aDSChannels::_updateField (OperationMode_t mode, int16_t value, uint8_t row) [protected]

Update value on LCD from given mode and row.

Parameters

<i>mode</i>	OperationMode_t : operation mode (SET/READ)
<i>value</i>	int16_t : value to display
<i>row</i>	uint8_t : LCD row

Returns

void

5.2.3.17 void aDSChannels::_wakeupFromStandby () [protected]

Wake up from standby mode.

Returns

void

5.2.3.18 template<typename T> bool aDSChannels::_write (T const v, int16_t & addr) [protected]

Template to write a value into EEPROM, at given address.

Parameters

<i>v</i>	T const : value
<i>addr</i>	int16_t& : start address

Returns

template <typename T> bool : true on write success, otherwise false

5.2.3.19 void aDSChannels::_writeMagicNumbers () [protected]

Write magic numbers into EEPROM.

Returns

void

5.2.3.20 aDSChannel::CalibrationData_t aDSChannels::getCalibrationValues (Channel_t chan)

Get calibration value for given channel.

Parameters

<i>chan</i>	Channel_t : channel
-------------	---------------------

Returns

[aDSChannel::CalibrationData_t](#) : calibration values

5.2.3.21 OperationMode_t aDSChannels::getOperationMode ()

Get current operation mode.

Returns

OperationMode_t : operation mode

5.2.3.22 void aDSChannels::incEncoderPosition (uint16_t v)

Increment or decrement encoder position.

Parameters

<i>v</i>	uint16_t : increment value (signed)
----------	-------------------------------------

Returns

void

5.2.3.23 bool aDSChannels::isInCalibration ()

Get calibration mode enableity.

Returns

bool : true if in calibration mode, otherwise false

5.2.3.24 bool aDSChannels::isInStandby ()

Get standby mode.

Returns

bool : true if in standby mode, otherwise false

5.2.3.25 bool aDSChannels::isJoinded ()

Get channel joining state.

Returns

bool : true if joinded, otherwise false

5.2.3.26 void aDSChannels::pingOperationMode ()

Reset timeout for OPERATION_MODE_SET mode.

Returns

void

5.2.3.27 void aDSChannels::restoreCalibrationValues ()

Restore calibration values from EEPROM.

Returns

void

5.2.3.28 void aDSChannels::saveCalibrationValues (Channel_t chan)

Save calibration value of given into EEPROM.

Parameters

<i>chan</i>	Channel_t : channel
-------------	---------------------

Returns

void

5.2.3.29 void aDSChannels::service ()

This member should be called often, it manage channels, LCD and EEPROM.

Returns

void

5.2.3.30 void aDSChannels::setCalibrationMode (bool enable)

Set calibration mode.

Parameters

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

Returns

void

5.2.3.31 void aDSChannels::setCalibrationValues (Channel_t chan, aDSChannel::CalibrationData_t cal)

Set calibration values for given channel.

Parameters

<i>chan</i>	Channel_t : channel
<i>cal</i>	aDSChannel::CalibrationData_t : calibration values

Returns

void

5.2.3.32 void aDSChannels::setFocusToNextChannel ()

Set the focus to the next channel, if any.

Returns

void

5.2.3.33 void aDSChannels::setOperationMode (OperationMode_t m)

Set operation mode (SET/READ)

Parameters

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

Returns

void

5.2.3.34 void aDSChannels::setup (uint8_t cols, uint8_t rows, uint8_t pwmChan1, uint8_t sensChan1, uint8_t ledChan1, uint8_t chkChan2, uint8_t pwmChan2, uint8_t sensChan2, uint8_t ledChan2)

Setup member function, should be called before any other member.

Parameters

<i>cols</i>	uint8_t : LCD number of columns
<i>rows</i>	uint8_t : LCD number of rows
<i>pwmChan1</i>	uint8_t : Channel 1 PWM pin

<i>sensChan1</i>	uint8_t : Channel 1 Temperature Sensor pin
<i>ledChan1</i>	uint8_t : Channel 1 LED pin
<i>chkChan2</i>	uint8_t : Channel 2 enability pin
<i>pwmChan2</i>	uint8_t : Channel 2 PWM pin
<i>sensChan2</i>	uint8_t : Channel 2 Temperature Sensor pin
<i>ledChan2</i>	uint8_t : Channel 2 LED pin

Returns

void

5.2.3.35 void aDSChannels::syncData (uint16_t bit)

Reset given bit inside bitfield, regardless of its state.

Parameters

<i>bit</i>	uint16_t : bit to reset
------------	-------------------------

Returns

void

5.2.3.36 void aDSChannels::toggleJoined ()

Toggle channels joining.

Returns

void

5.2.3.37 void aDSChannels::toggleStandbyMode ()

Toggle standby mode.

Returns

void

5.2.3.38 void aDSChannels::updateOperationMode ()

Update operation mode.

If operation mode is currently set to OPERATION_MODE_SET, and OPERATION_SET_TIMEOUT timeout is triggered, operation mode will be switched to OPERATION_MODE_READ

Returns

void

5.2.4 Member Data Documentation**5.2.4.1 const uint16_t aDSChannels::DATA_CHANNEL1_LED_STATE = 1 << 5 [static]**

Bitfield: LED state of channel 1 has changed.

5.2.4.2 `const uint16_t aDSChannels::DATA_CHANNEL1_TEMP_READ = 1 << 4` [static]

Bitfield: Readed temperature of channel 1 has changed.

5.2.4.3 `const uint16_t aDSChannels::DATA_CHANNEL1_TEMP_SET = 1 << 3` [static]

Bitfield: Target temperature of channel 1 has changed.

5.2.4.4 `const uint16_t aDSChannels::DATA_CHANNEL2_ENABLED = 1` [static]

Bitfield: Channel 2 is enabled.

5.2.4.5 `const uint16_t aDSChannels::DATA_CHANNEL2_LED_STATE = 1 << 8` [static]

Bitfield: LED state of channel 1 has changed.

5.2.4.6 `const uint16_t aDSChannels::DATA_CHANNEL2_TEMP_READ = 1 << 7` [static]

Bitfield: Readed temperature of channel 1 has changed.

5.2.4.7 `const uint16_t aDSChannels::DATA_CHANNEL2_TEMP_SET = 1 << 6` [static]

Bitfield: Target temperature of channel 1 has changed.

5.2.4.8 `const uint16_t aDSChannels::DATA_CHANNELS_JOINED = 1 << 1` [static]

Bitfield: Channel 1 & 2 are joined.

5.2.4.9 `const uint16_t aDSChannels::DATA_DISPLAY = 1 << 9` [static]

Bitfield: Display should be refreshed.

5.2.4.10 `const uint16_t aDSChannels::DATA_DISPLAY_STANDBY = 1 << 11` [static]

Bitfield: Standby state has changed.

5.2.4.11 `const uint16_t aDSChannels::DATA_FOCUS = 1 << 12` [static]

Bitfield: Focus has changed.

5.2.4.12 `const uint16_t aDSChannels::DATA_IN_CALIBRATION = 1 << 13` [static]

Bitfield: in Calibration.

5.2.4.13 `const uint16_t aDSChannels::DATA_OPERATION = 1 << 2` [static]

Bitfield: Operation mode has changed.

5.2.4.14 `const uint16_t aDSChannels::DATA_STANDBY = 1 << 10` `[static]`

Bitfield: Standby state.

5.2.4.15 `const unsigned long aDSChannels::DISPLAY_UPDATE_RATE = 200` `[static]`

Display update rate, in ms.

5.2.4.16 `const int16_t aDSChannels::EEPROM_ADDR_CALIBRATION_CHAN_1 = EEPROM_ADDR_TEMP_CHANNEL_T-
WO + EEPROM_TEMP_SIZE` `[static]`, `[protected]`

EEPROM start offset for Channel 1 calibration values.

5.2.4.17 `const int16_t aDSChannels::EEPROM_ADDR_CALIBRATION_CHAN_2 = EEPROM_AD-
DR_CALIBRATION_CHAN_1 + EEPROM_CALIBRATION_SIZE` `[static]`,
`[protected]`

EEPROM start offset for Channel 2 calibration values.

5.2.4.18 `const int16_t aDSChannels::EEPROM_ADDR_CHANNEL_JOINED = EEPROM_STORAGE_STARTING + 1`
`[static]`, `[protected]`

Channels are joined.

5.2.4.19 `const int16_t aDSChannels::EEPROM_ADDR_MAGIC = 0` `[static]`, `[protected]`

EEPROM offset storage start for magic numbers (0xDEAD)

5.2.4.20 `const int16_t aDSChannels::EEPROM_ADDR_TEMP_CHANNEL_ONE = EEPROM_ADDR_CHANNEL_JOINED +
EEPROM_TEMP_SIZE` `[static]`, `[protected]`

Target temp for Channel 1.

5.2.4.21 `const int16_t aDSChannels::EEPROM_ADDR_TEMP_CHANNEL_TWO = EEPROM_ADDR_TEMP_CHANNEL_O-
NE + EEPROM_TEMP_SIZE` `[static]`, `[protected]`

Target temp for Channel 2.

5.2.4.22 `const int16_t aDSChannels::EEPROM_CALIBRATION_SIZE = (sizeof(float) * 2) + sizeof(uint8_t)` `[static]`,
`[protected]`

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

5.2.4.23 `const int16_t aDSChannels::EEPROM_STORAGE_STARTING = 5` `[static]`, `[protected]`

EEPROM starting address for program datas.

5.2.4.24 `const int16_t aDSChannels::EEPROM_TEMP_SIZE = sizeof(uint16_t) + sizeof(uint8_t)` `[static]`,
`[protected]`

EEPROM temperature size (temperature + crc)

5.2.4.25 **aDSChannel** aDSChannels::m_channels[CHANNEL_MAX] [private]

5.2.4.26 **uint16_t** aDSChannels::m_dats [private]

5.2.4.27 **bool** aDSChannels::m_isValidEEPROM [private]

5.2.4.28 **unsigned long** aDSChannels::m_lastTempChange [private]

5.2.4.29 **LiquidCrystal** aDSChannels::m_lcd [private]

5.2.4.30 **uint8_t** aDSChannels::m_lcdCols [private]

5.2.4.31 **uint8_t** aDSChannels::m_lcdRows [private]

5.2.4.32 **unsigned long** aDSChannels::m_nextDisplayUpdate [private]

5.2.4.33 **unsigned long** aDSChannels::m_nextMeasureUpdate [private]

5.2.4.34 **OperationMode_t** aDSChannels::m_operationMode [private]

5.2.4.35 **unsigned long** aDSChannels::m_operationTick [private]

5.2.4.36 **bool** aDSChannels::m_storedToEEPROM [private]

5.2.4.37 **const unsigned long** aDSChannels::MEASURE_UPDATE_RATE = 200 [static]

Measurement (for [aDSChannel](#)) rate, in ms.

5.2.4.38 **const uint8_t** aDSChannels::OFFSET_MARKER_LEFT = 0 [static]

Column LCD offset for left marker '['.

5.2.4.39 **const uint8_t** aDSChannels::OFFSET_MARKER_RIGHT = 10 [static]

Column LCD offset for right marker ']'.

5.2.4.40 **const uint8_t** aDSChannels::OFFSET_VALUE = 2 [static]

Value column LCD offset.

5.2.4.41 **const unsigned long** aDSChannels::OPERATION_SET_TIMEOUT = 3000 [static]

Automatic toggle settings->reading timeout (3 seconds), in ms.

5.2.4.42 **const unsigned long** aDSChannels::TEMP_SETTING_INACTIVITY = 60000 [static]

Timeout in ms, after which the new target temperature will be stored in the EEPROM.

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

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

5.3 aDSEngine Class Reference

```
#include <aDSEngine.h>
```

Public Member Functions

- [aDSEngine \(\)](#)
aDSEngine constructor
- virtual [~aDSEngine \(\)](#)
aDSEngine destructor
- void [setup \(\)](#)
Setup member function, should be called before any other member.
- void [run \(\)](#)
Main loop.

Protected Member Functions

- void [_handleSerialInput \(\)](#)
Handle serial input, in calibration mode only.

Private Attributes

- [aDSChannels m_channels](#)
- ClickEncoder [m_encoder](#)
- uint16_t [m_datas](#)
- uint8_t [m_RXbuffer](#) [RXBUFFER_MAXLEN]
USB rx buffer.
- uint8_t [m_RXoffset](#)
USB rx buffer offset counter.
- unsigned long [m_serialInputTick](#)

Static Private Attributes

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

5.3.1 Constructor & Destructor Documentation

5.3.1.1 aDSEngine::aDSEngine ()

[aDSEngine](#) constructor

5.3.1.2 aDSEngine::~~aDSEngine () [virtual]

[aDSEngine](#) destructor

5.3.2 Member Function Documentation

5.3.2.1 void aDSEngine::_handleSerialInput () [protected]

Handle serial input, in calibration mode only.

Returns

void

5.3.2.2 void aDSEngine::run ()

Main loop.

Returns

void

5.3.2.3 void aDSEngine::setup ()

Setup member function, should be called before any other member.

Returns

void

5.3.3 Member Data Documentation

5.3.3.1 aDSChannels aDSEngine::m_channels [private]

5.3.3.2 uint16_t aDSEngine::m_datas [private]

5.3.3.3 ClickEncoder aDSEngine::m_encoder [private]

5.3.3.4 uint8_t aDSEngine::m_RXbuffer[RXBUFFER_MAXLEN] [private]

USB rx buffer.

5.3.3.5 uint8_t aDSEngine::m_RXoffset [private]

USB rx buffer offset counter.

5.3.3.6 unsigned long aDSEngine::m_serialInputTick [private]

5.3.3.7 const uint8_t aDSEngine::RXBUFFER_MAXLEN = 64 [static], [private]

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

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

5.4 aDSChannel::aPin_t Struct Reference

Our pin structure.

```
#include <aDSEngine.h>
```

Public Attributes

- [uint8_t pin](#)
"Arduino" pin
- [uint8_t timer](#)
Timer of the pin.
- [uint8_t mask](#)
Bit mask of the pin.
- [uint8_t port](#)
Port of the pin.
- [volatile uint8_t * outputRegister](#)
Output register of the pin.

5.4.1 Detailed Description

Our pin structure.

5.4.2 Member Data Documentation

5.4.2.1 [uint8_t aDSChannel::aPin_t::mask](#)

Bit mask of the pin.

5.4.2.2 [volatile uint8_t* aDSChannel::aPin_t::outputRegister](#)

Output register of the pin.

5.4.2.3 [uint8_t aDSChannel::aPin_t::pin](#)

"Arduino" pin

5.4.2.4 [uint8_t aDSChannel::aPin_t::port](#)

Port of the pin.

5.4.2.5 [uint8_t aDSChannel::aPin_t::timer](#)

Timer of the pin.

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

- [aDSEngine.h](#)

Chapter 6

File Documentation

6.1 aDSEngine.cpp File Reference

```
#include "aDSEngine.h"  
#include <wiring_private.h>
```

Functions

- static int8_t [getNumericalLength](#) (int16_t n)
Return numerical character length of argument.
- void [timer1ISR](#) (void)
Timer1 class ISR function.

Variables

- static const uint8_t [DIGIT_WIDTH](#) = 3
Max numerical length of temperature (used with big digits)
- static const uint8_t [_glyphs](#)[][8] [PROGMEM](#)
LCD glyphs (for big digits and LED)
- static const uint8_t [_bigDigitsTop](#) [11][[DIGIT_WIDTH](#)]
_glyphs[] offsets
- static const uint8_t [_bigDigitsBottom](#) [11][[DIGIT_WIDTH](#)]
_glyphs[] offsets
- ClickEncoder * [pEncoder](#) = NULL
Global pointer to ClickEncoder object, used inside [timer1ISR\(\)](#) function.

6.1.1 Detailed Description

Copyright

Copyright (C) 2015 F1RMB, Daniel Caujolle-Bert f1rmb.daniel@gmail.com

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

6.1.2 Function Documentation

6.1.2.1 static int8_t getNumericalLength (int16_t n) [static]

Return numerical character length of argument.

Parameters

<i>n</i>	int16_t : value to get length from
----------	------------------------------------

Returns

int8_t : length

6.1.2.2 void timer1ISR (void)

Timer1 class ISR function.

This function is periodically called from Timer1 class, which handles encoder events

Returns

void

6.1.3 Variable Documentation

6.1.3.1 const uint8_t _bigDigitsBottom[11][DIGIT_WIDTH] [static]

Initial value:

```
=
{
    { 3, 1, 3 },
    { 1, 3, 1 },
    { 3, 1, 1 },
    { 1, 1, 3 },
    { 32, 32, 3 },
    { 1, 1, 3 },
    { 3, 1, 3 },
    { 32, 32, 3 },
    { 3, 1, 3 },
    { 1, 1, 3 },
    { 32, 32, 32 }
}
```

_glyphs[] offsets

0..9 + ' ' bottom characters matrix

6.1.3.2 `const uint8_t _bigDigitsTop[11][DIGIT_WIDTH]` `[static]`

Initial value:

```
=
{
    { 3, 0, 3 },
    { 0, 3, 32 },
    { 2, 2, 3 },
    { 0, 2, 3 },
    { 3, 1, 3 },
    { 3, 2, 2 },
    { 3, 2, 2 },
    { 0, 0, 3 },
    { 3, 2, 3 },
    { 3, 2, 3 },
    { 32, 32, 32 }
}
```

`_glyphs[]` offsets

0..9 + ' ' top characters matrix

6.1.3.3 `const uint8_t DIGIT_WIDTH = 3` `[static]`

Max numerical length of temperature (used with big digits)

le processus de calibration est necessaire à la première utilisation de la station de soudage Pour mettre en fonction ce processus, il faut appuyer et maintenir le bouton de l'encoder tout en allumant la station. Un indicateur de mise en fonction de ce mode ('CAL') sera affiché sur la partie droite de la première ligne de l'afficheur LCD. En lieu et place de la temperature lue, pour chaque canal, la valeur ADC sera affichée. Cette valeur est utile au processus de calibration.

6.1.3.4 `ClickEncoder* pEncoder = NULL`

Global pointer to ClickEncoder object, used inside `timer1ISR()` function.

6.1.3.5 `const uint8_t _glyphs[][8] PROGMEM` `[static]`

LCD glyphs (for big digits and LED)

6.2 aDSEngine.h File Reference

```
#include <Arduino.h>
#include <LiquidCrystal.h>
#include <EEPROM.h>
#include "TimerOne.h"
#include "ClickEncoder.h"
```

Classes

- class [aDSChannel](#)
aDSChannel class
- struct [aDSChannel::CalibrationData_t](#)
Calibration values.
- struct [aDSChannel::aPin_t](#)
Our pin structure.

- class [aDSChannels](#)
- union [aDSChannels::_eepromCalibrationValue_t](#)
Union to manipulate float/uint8_t [] calibration values.
- class [aDSEngine](#)

Macros

- `#define SIMU 1`
- `#define IS_DATA_ENABLED(bit) ((m_datas & bit))`

Enumerations

- enum [OperationMode_t](#) { OPERATION_MODE_READ, OPERATION_MODE_SET, OPERATION_MODE_UNKNOWN }
- Operation Mode enumeration.*

Variables

- static const uint8_t [CHANNEL2_ENABLE_PIN](#) = 13
Pin to check from if channel 2 is wired.
- static const uint8_t [LCD_RS_PIN](#) = 7
LCD RS pin.
- static const uint8_t [LCD_ENABLE_PIN](#) = 8
LCD Enable pin.
- static const uint8_t [LCD_D4_PIN](#) = 9
LCD D4 pin.
- static const uint8_t [LCD_D5_PIN](#) = 10
LCD D5 pin.
- static const uint8_t [LCD_D6_PIN](#) = 11
LCD D6 pin.
- static const uint8_t [LCD_D7_PIN](#) = 12
LCD D7 pin.
- static const uint8_t [LCD_COLS](#) = 16
LCD columns.
- static const uint8_t [LCD_ROWS](#) = 2
LCD rows.
- static const uint8_t [ENCODER_A_PIN](#) = 2
Encoder A pin.
- static const uint8_t [ENCODER_B_PIN](#) = 3
Encoder B pin.
- static const uint8_t [ENCODER_PB_PIN](#) = 4
Encoder push button pin.
- static const uint8_t [ENCODER_STEPS_PER_NOTCH](#) = 4
Number of steps per notch (indent)
- static const uint8_t [PWM_CHANNEL1_PIN](#) = 5
PWM pin of channel 1.
- static const uint8_t [PWM_CHANNEL2_PIN](#) = 6
PWM pin of channel 2.
- static const uint8_t [TEMP_SENSOR_CHANNEL1_PIN](#) = A1
Temp sensor pin of channel 1.

- static const uint8_t `TEMP_SENSOR_CHANNEL2_PIN` = A0
Temp sensor pin of channel 2.
- static const uint8_t `LED_CHANNEL1_PIN` = A2
LED pin of channel 1.
- static const uint8_t `LED_CHANNEL2_PIN` = A3
LED pin of channel 2.
- static const uint8_t `PROGRAM_VERSION_MAJOR` = 1
Major program version.
- static const uint8_t `PROGRAM_VERSION_MINOR` = 2
Minor program version.

6.2.1 Detailed Description

Copyright

Copyright (C) 2015 F1RMB, Daniel Caujolle-Bert flrmb.daniel@gmail.com

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 flrmb.daniel@gmail.com

6.2.2 Macro Definition Documentation

6.2.2.1 `#define IS_DATA_ENABLED(bit) ((m_datas & bit))`

6.2.2.2 `#define SIMU 1`

6.2.3 Enumeration Type Documentation

6.2.3.1 `enum OperationMode_t`

Operation Mode enumeration.

Enumerator

`OPERATION_MODE_READ` Reading values.

`OPERATION_MODE_SET` Settings values.

`OPERATION_MODE_UNKNOWN` Unset (internal)

6.2.4 Variable Documentation

6.2.4.1 `const uint8_t CHANNEL2_ENABLE_PIN = 13` `[static]`

Pin to check from if channel 2 is wired.

6.2.4.2 `const uint8_t ENCODER_A_PIN = 2` `[static]`

Encoder A pin.

6.2.4.3 `const uint8_t ENCODER_B_PIN = 3` `[static]`

Encoder B pin.

6.2.4.4 `const uint8_t ENCODER_PB_PIN = 4` `[static]`

Encoder push button pin.

6.2.4.5 `const uint8_t ENCODER_STEPS_PER_NOTCH = 4` `[static]`

Number of steps per notch (indent)

6.2.4.6 `const uint8_t LCD_COLS = 16` `[static]`

LCD columns.

6.2.4.7 `const uint8_t LCD_D4_PIN = 9` `[static]`

LCD D4 pin.

6.2.4.8 `const uint8_t LCD_D5_PIN = 10` `[static]`

LCD D5 pin.

6.2.4.9 `const uint8_t LCD_D6_PIN = 11` `[static]`

LCD D6 pin.

6.2.4.10 `const uint8_t LCD_D7_PIN = 12` `[static]`

LCD D7 pin.

6.2.4.11 `const uint8_t LCD_ENABLE_PIN = 8` `[static]`

LCD Enable pin.

6.2.4.12 `const uint8_t LCD_ROWS = 2` `[static]`

LCD rows.

6.2.4.13 `const uint8_t LCD_RS_PIN = 7` `[static]`

LCD RS pin.

6.2.4.14 `const uint8_t LED_CHANNEL1_PIN = A2` `[static]`

LED pin of channel 1.

6.2.4.15 `const uint8_t LED_CHANNEL2_PIN = A3` `[static]`

LED pin of channel 2.

6.2.4.16 `const uint8_t PROGRAM_VERSION_MAJOR = 1` `[static]`

Major program version.

6.2.4.17 `const uint8_t PROGRAM_VERSION_MINOR = 2` `[static]`

Minor program version.

6.2.4.18 `const uint8_t PWM_CHANNEL1_PIN = 5` `[static]`

PWM pin of channel 1.

6.2.4.19 `const uint8_t PWM_CHANNEL2_PIN = 6` `[static]`

PWM pin of channel 2.

6.2.4.20 `const uint8_t TEMP_SENSOR_CHANNEL1_PIN = A1` `[static]`

Temp sensor pin of channel 1.

6.2.4.21 `const uint8_t TEMP_SENSOR_CHANNEL2_PIN = A0` `[static]`

Temp sensor pin of channel 2.

6.3 CDC.cpp File Reference

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

6.4 HardwareSerial.cpp File Reference

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

6.5 HID.cpp File Reference

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

6.6 IPAddress.cpp File Reference

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

6.7 libraries.cpp File Reference

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

6.8 main.cpp File Reference

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

6.9 new.cpp File Reference

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

6.10 Print.cpp File Reference

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

6.11 sketch.cpp File Reference

```
#include <Arduino.h>  
#include <LiquidCrystal.h>  
#include "aDSEngine.h"
```

Functions

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

Variables

- [aDSEngine](#) * [engine](#)

6.11.1 Function Documentation

6.11.1.1 void loop ()

6.11.1.2 void setup ()

6.11.2 Variable Documentation

6.11.2.1 aDSEngine* engine

6.12 Stream.cpp File Reference

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

6.13 Tone.cpp File Reference

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

6.14 USBCore.cpp File Reference

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

6.15 WInterrupts.c File Reference

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

6.16 wiring.c File Reference

```
#include "wiring_private.h"
```

Macros

- #define MICROSECONDS_PER_TIMER0_OVERFLOW (clockCyclesToMicroseconds(PRESCALE_FACTOR * 256))
- #define MILLIS_INC (MICROSECONDS_PER_TIMER0_OVERFLOW / 1000)
- #define FRACT_INC ((MICROSECONDS_PER_TIMER0_OVERFLOW % 1000) >> 3)
- #define FRACT_MAX (1000 >> 3)

Functions

- if (f >= FRACT_MAX)
- unsigned long millis ()
- unsigned long micros ()
- void delay (unsigned long ms)
- void delayMicroseconds (unsigned int us)
- void init ()

Variables

- volatile unsigned long `timer0_overflow_count` = 0
- volatile unsigned long `timer0_millis` = 0
- static unsigned char `timer0_fract` = 0
- unsigned char `f` = `timer0_fract`
- `m` = `MILLIS_INC`

6.16.1 Macro Definition Documentation

6.16.1.1 `#define FRACT_INC ((MICROSECONDS_PER_TIMER0_OVERFLOW % 1000) >> 3)`

6.16.1.2 `#define FRACT_MAX (1000 >> 3)`

6.16.1.3 `#define MICROSECONDS_PER_TIMER0_OVERFLOW (clockCyclesToMicroseconds(PRESCALE_FACTOR * 256))`

6.16.1.4 `#define MILLIS_INC (MICROSECONDS_PER_TIMER0_OVERFLOW / 1000)`

6.16.2 Function Documentation

6.16.2.1 `void delay (unsigned long ms)`

6.16.2.2 `void delayMicroseconds (unsigned int us)`

6.16.2.3 `if (f >= FRACT_MAX)`

6.16.2.4 `void init ()`

6.16.2.5 `unsigned long micros ()`

6.16.2.6 `unsigned long millis ()`

6.16.3 Variable Documentation

6.16.3.1 `f = timer0_fract`

6.16.3.2 `m = MILLIS_INC`

6.16.3.3 `timer0_fract = 0` `[static]`

6.16.3.4 `timer0_millis = 0`

6.16.3.5 `timer0_overflow_count = 0`

6.17 wiring_analog.c File Reference

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

6.18 wiring_digital.c File Reference

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


6.19 wiring_pulse.c File Reference

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

6.20 wiring_shift.c File Reference

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

6.21 WMath.cpp File Reference

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

6.22 WString.cpp File Reference

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

Index

- ~aDSChannel
 - aDSChannel, [11](#)
- ~aDSChannels
 - aDSChannels, [21](#)
- ~aDSEngine
 - aDSEngine, [33](#)
- _analogRead
 - aDSChannel, [12](#)
- _analogWrite
 - aDSChannel, [12](#)
- _backupCalibrationFromEEPROM
 - aDSChannels, [21](#)
- _bigDigitsBottom
 - aDSEngine.cpp, [38](#)
- _bigDigitsTop
 - aDSEngine.cpp, [38](#)
- _checkForMagicNumbers
 - aDSChannels, [22](#)
- _clearValue
 - aDSChannels, [22](#)
- _crc8
 - aDSChannels, [22](#)
- _digitalRead
 - aDSChannel, [12](#)
- _digitalWrite
 - aDSChannel, [12](#)
- _displayBigDigit
 - aDSChannels, [22](#)
- _displayBigDigits
 - aDSChannels, [23](#)
- _enableData
 - aDSChannels, [23](#)
- _enableDataCheck
 - aDSChannels, [23](#)
- _getTempFromEEPROM
 - aDSChannels, [23](#)
- _handleSerialInput
 - aDSEngine, [34](#)
- _read
 - aDSChannels, [24](#)
- _restoreCalibrationFromEEPROM
 - aDSChannels, [24](#)
- _scissor
 - aDSChannels, [24](#)
- _setTempToEEPROM
 - aDSChannels, [24](#)
- _showBanner
 - aDSChannels, [25](#)
- _turnOffPWM

- aDSChannel, [12](#)
- _updateDisplay
 - aDSChannels, [25](#)
- _updateField
 - aDSChannels, [25](#)
- _wakeupFromStandby
 - aDSChannels, [25](#)
- _write
 - aDSChannels, [25](#)
- _writeMagicNumbers
 - aDSChannels, [26](#)
- aDSChannel
 - HEATING_STATE_COOLING, [11](#)
 - HEATING_STATE_HEATING, [11](#)
 - HEATING_STATE_REACHED, [11](#)
 - HEATING_STATE_STANDBY, [11](#)
- aDSChannels
 - CHANNEL_MAX, [21](#)
 - CHANNEL_ONE, [21](#)
 - CHANNEL_TWO, [21](#)
- aDSEngine.h
 - OPERATION_MODE_READ, [41](#)
 - OPERATION_MODE_SET, [41](#)
 - OPERATION_MODE_UNKNOWN, [41](#)
- ADC_TO_TEMP_GAIN
 - aDSChannel, [16](#)
- aDSChannel, [9](#)
 - ~aDSChannel, [11](#)
 - _analogRead, [12](#)
 - _analogWrite, [12](#)
 - _digitalRead, [12](#)
 - _digitalWrite, [12](#)
 - _turnOffPWM, [12](#)
 - ADC_TO_TEMP_GAIN, [16](#)
 - aDSChannel, [11](#)
 - aDSChannel, [11](#)
 - BLINK_UPDATE_RATE, [16](#)
 - CNTRL_GAIN, [16](#)
 - getADCValue, [13](#)
 - getCalibration, [13](#)
 - getHeatState, [13](#)
 - getLEDState, [13](#)
 - getStandbyMode, [13](#)
 - getTemperature, [13](#)
 - hasFocus, [14](#)
 - HeatingState_t, [11](#)
 - isTempHasChanged, [14](#)
 - m_adcValue, [16](#)
 - m_blinkStandby, [16](#)

- m_cal, 16
- m_channel, 16
- m_currentTemp, 16
- m_hasFocus, 16
- m_heatState, 16
- m_inStandby, 16
- m_ledPin, 16
- m_ledState, 16
- m_nextBlink, 16
- m_nextLowering, 16
- m_nextPass, 16
- m_nextTempStep, 16
- m_pwmPin, 16
- m_pwmValue, 16
- m_ref, 16
- m_sensorPin, 17
- m_targetTemp, 17
- m_tempHasChanged, 17
- PWM_MAX_VALUE, 17
- service, 14
- setCalibration, 14
- setFocus, 14
- setStandbyMode, 14
- setTemperature, 15
- setup, 15
- syncTempChange, 15
- TEMPERATURE_MAX, 17
- TEMPERATURE_MIN, 17
- updateLEDState, 15
- aDSChannel::aPin_t, 35
 - mask, 35
 - outputRegister, 35
 - pin, 35
 - port, 35
 - timer, 35
- aDSChannels, 17
 - ~aDSChannels, 21
 - _backupCalibrationFromEEPROM, 21
 - _checkForMagicNumbers, 22
 - _clearValue, 22
 - _crc8, 22
 - _displayBigDigit, 22
 - _displayBigDigits, 23
 - _enableData, 23
 - _enableDataCheck, 23
 - _getTempFromEEPROM, 23
 - _read, 24
 - _restoreCalibrationFromEEPROM, 24
 - _scissor, 24
 - _setTempToEEPROM, 24
 - _showBanner, 25
 - _updateDisplay, 25
 - _updateField, 25
 - _wakeupFromStandby, 25
 - _write, 25
 - _writeMagicNumbers, 26
 - aDSChannels, 21
 - aDSChannels, 21
- Channel_t, 21
- DATA_DISPLAY, 30
- DATA_FOCUS, 30
- DATA_OPERATION, 30
- DATA_STANDBY, 30
- EEPROM_ADDR_MAGIC, 31
- EEPROM_TEMP_SIZE, 31
- getCalibrationValues, 26
- getOperationMode, 26
- incEncoderPosition, 26
- isInCalibration, 26
- isInStandby, 26
- isJoined, 27
- m_channels, 31
- m_dats, 32
- m_isValidEEPROM, 32
- m_lastTempChange, 32
- m_lcd, 32
- m_lcdCols, 32
- m_lcdRows, 32
- m_nextDisplayUpdate, 32
- m_nextMeasureUpdate, 32
- m_operationMode, 32
- m_operationTick, 32
- m_storedToEEPROM, 32
- OFFSET_VALUE, 32
- pingOperationMode, 27
- restoreCalibrationValues, 27
- saveCalibrationValues, 27
- service, 27
- setCalibrationMode, 27
- setCalibrationValues, 28
- setFocusToNextChannel, 28
- setOperationMode, 28
- setup, 28
- syncData, 29
- toggleJoined, 29
- toggleStandbyMode, 29
- updateOperationMode, 29
- aDSEngine, 33
 - ~aDSEngine, 33
 - _handleSerialInput, 34
 - aDSEngine, 33
 - aDSEngine, 33
 - m_RXbuffer, 34
 - m_RXoffset, 34
 - m_channels, 34
 - m_dats, 34
 - m_encoder, 34
 - m_serialInputTick, 34
 - RXBUFFER_MAXLEN, 34
 - run, 34
 - setup, 34
- aDSEngine.cpp, 37
 - _bigDigitsBottom, 38
 - _bigDigitsTop, 38
 - DIGIT_WIDTH, 39
 - getNumericalLength, 38

- pEncoder, 39
- PROGMEM, 39
- timer1ISR, 38
- aDSEngine.h, 39
 - ENCODER_A_PIN, 41
 - ENCODER_B_PIN, 42
 - ENCODER_PB_PIN, 42
 - IS_DATA_ENABLED, 41
 - LCD_COLS, 42
 - LCD_D4_PIN, 42
 - LCD_D5_PIN, 42
 - LCD_D6_PIN, 42
 - LCD_D7_PIN, 42
 - LCD_ENABLE_PIN, 42
 - LCD_ROWS, 42
 - LCD_RS_PIN, 42
 - LED_CHANNEL1_PIN, 42
 - LED_CHANNEL2_PIN, 43
 - OperationMode_t, 41
 - PWM_CHANNEL1_PIN, 43
 - PWM_CHANNEL2_PIN, 43
 - SIMU, 41
- BLINK_UPDATE_RATE
 - aDSChannel, 16
- CHANNEL_MAX
 - aDSChannels, 21
- CHANNEL_ONE
 - aDSChannels, 21
- CHANNEL_TWO
 - aDSChannels, 21
- CDC.cpp, 43
- CHANNEL2_ENABLE_PIN
 - aDSEngine.h, 41
- CNTRL_GAIN
 - aDSChannel, 16
- Channel_t
 - aDSChannels, 21
- DATA_DISPLAY
 - aDSChannels, 30
- DATA_FOCUS
 - aDSChannels, 30
- DATA_OPERATION
 - aDSChannels, 30
- DATA_STANDBY
 - aDSChannels, 30
- DIGIT_WIDTH
 - aDSEngine.cpp, 39
- delay
 - wiring.c, 46
- delayMicroseconds
 - wiring.c, 46
- EEPROM_ADDR_MAGIC
 - aDSChannels, 31
- EEPROM_TEMP_SIZE
 - aDSChannels, 31
- ENCODER_A_PIN
 - aDSEngine.h, 41
- ENCODER_B_PIN
 - aDSEngine.h, 42
- ENCODER_PB_PIN
 - aDSEngine.h, 42
- engine
 - sketch.cpp, 45
- f
 - wiring.c, 46
- FRACT_INC
 - wiring.c, 46
- FRACT_MAX
 - wiring.c, 46
- getADCValue
 - aDSChannel, 13
- getCalibration
 - aDSChannel, 13
- getCalibrationValues
 - aDSChannels, 26
- getHeatState
 - aDSChannel, 13
- getLEDState
 - aDSChannel, 13
- getNumericalLength
 - aDSEngine.cpp, 38
- getOperationMode
 - aDSChannels, 26
- getStandbyMode
 - aDSChannel, 13
- getTemperature
 - aDSChannel, 13
- HEATING_STATE_COOLING
 - aDSChannel, 11
- HEATING_STATE_HEATING
 - aDSChannel, 11
- HEATING_STATE_REACHED
 - aDSChannel, 11
- HEATING_STATE_STANDBY
 - aDSChannel, 11
- HID.cpp, 43
- HardwareSerial.cpp, 43
- hasFocus
 - aDSChannel, 14
- HeatingState_t
 - aDSChannel, 11
- IPAddress.cpp, 44
- IS_DATA_ENABLED
 - aDSEngine.h, 41
- if
 - wiring.c, 46
- incEncoderPosition
 - aDSChannels, 26
- init
 - wiring.c, 46

- isInCalibration
 - aDSChannels, 26
- isInStandby
 - aDSChannels, 26
- isJoinded
 - aDSChannels, 27
- isTempHasChanged
 - aDSChannel, 14
- LCD_COLS
 - aDSEngine.h, 42
- LCD_D4_PIN
 - aDSEngine.h, 42
- LCD_D5_PIN
 - aDSEngine.h, 42
- LCD_D6_PIN
 - aDSEngine.h, 42
- LCD_D7_PIN
 - aDSEngine.h, 42
- LCD_ENABLE_PIN
 - aDSEngine.h, 42
- LCD_ROWS
 - aDSEngine.h, 42
- LCD_RS_PIN
 - aDSEngine.h, 42
- LED_CHANNEL1_PIN
 - aDSEngine.h, 42
- LED_CHANNEL2_PIN
 - aDSEngine.h, 43
- libraries.cpp, 44
- loop
 - sketch.cpp, 44
- m
 - wiring.c, 46
- m_RXbuffer
 - aDSEngine, 34
- m_RXoffset
 - aDSEngine, 34
- m_adcValue
 - aDSChannel, 16
- m_blinkStandby
 - aDSChannel, 16
- m_cal
 - aDSChannel, 16
- m_channel
 - aDSChannel, 16
- m_channels
 - aDSChannels, 31
 - aDSEngine, 34
- m_currentTemp
 - aDSChannel, 16
- m_datas
 - aDSChannels, 32
 - aDSEngine, 34
- m_encoder
 - aDSEngine, 34
- m_hasFocus
 - aDSChannel, 16
- m_heatState
 - aDSChannel, 16
- m_inStandby
 - aDSChannel, 16
- m_isValidEEPROM
 - aDSChannels, 32
- m_lastTempChange
 - aDSChannels, 32
- m_lcd
 - aDSChannels, 32
- m_lcdCols
 - aDSChannels, 32
- m_lcdRows
 - aDSChannels, 32
- m_ledPin
 - aDSChannel, 16
- m_ledState
 - aDSChannel, 16
- m_nextBlink
 - aDSChannel, 16
- m_nextDisplayUpdate
 - aDSChannels, 32
- m_nextLowering
 - aDSChannel, 16
- m_nextMeasureUpdate
 - aDSChannels, 32
- m_nextPass
 - aDSChannel, 16
- m_nextTempStep
 - aDSChannel, 16
- m_operationMode
 - aDSChannels, 32
- m_operationTick
 - aDSChannels, 32
- m_pwmPin
 - aDSChannel, 16
- m_pwmValue
 - aDSChannel, 16
- m_ref
 - aDSChannel, 16
- m_sensorPin
 - aDSChannel, 17
- m_serialInputTick
 - aDSEngine, 34
- m_storedToEEPROM
 - aDSChannels, 32
- m_targetTemp
 - aDSChannel, 17
- m_tempHasChanged
 - aDSChannel, 17
- MILLIS_INC
 - wiring.c, 46
- main.cpp, 44
- mask
 - aDSChannel::aPin_t, 35
- micros
 - wiring.c, 46
- millis

- wiring.c, 46
- new.cpp, 44
- OPERATION_MODE_READ
 - aDSEngine.h, 41
- OPERATION_MODE_SET
 - aDSEngine.h, 41
- OPERATION_MODE_UNKNOWN
 - aDSEngine.h, 41
- OFFSET_MARKER_LEFT
 - aDSChannels, 32
- OFFSET_VALUE
 - aDSChannels, 32
- OperationMode_t
 - aDSEngine.h, 41
- outputRegister
 - aDSChannel::aPin_t, 35
- pEncoder
 - aDSEngine.cpp, 39
- PROGMEM
 - aDSEngine.cpp, 39
- PWM_CHANNEL1_PIN
 - aDSEngine.h, 43
- PWM_CHANNEL2_PIN
 - aDSEngine.h, 43
- PWM_MAX_VALUE
 - aDSChannel, 17
- pin
 - aDSChannel::aPin_t, 35
- pingOperationMode
 - aDSChannels, 27
- port
 - aDSChannel::aPin_t, 35
- Print.cpp, 44
- RXBUFFER_MAXLEN
 - aDSEngine, 34
- restoreCalibrationValues
 - aDSChannels, 27
- run
 - aDSEngine, 34
- SIMU
 - aDSEngine.h, 41
- saveCalibrationValues
 - aDSChannels, 27
- service
 - aDSChannel, 14
 - aDSChannels, 27
- setCalibration
 - aDSChannel, 14
- setCalibrationMode
 - aDSChannels, 27
- setCalibrationValues
 - aDSChannels, 28
- setFocus
 - aDSChannel, 14
- setFocusToNextChannel
 - aDSChannels, 28
- setOperationMode
 - aDSChannels, 28
- setStandbyMode
 - aDSChannel, 14
- setTemperature
 - aDSChannel, 15
- setup
 - aDSChannel, 15
 - aDSChannels, 28
 - aDSEngine, 34
 - sketch.cpp, 44
- sketch.cpp, 44
 - engine, 45
 - loop, 44
 - setup, 44
- Stream.cpp, 45
- syncData
 - aDSChannels, 29
- syncTempChange
 - aDSChannel, 15
- TEMPERATURE_MAX
 - aDSChannel, 17
- TEMPERATURE_MIN
 - aDSChannel, 17
- TEMPERATURE_STANDBY
 - aDSChannel, 17
- timer
 - aDSChannel::aPin_t, 35
- timer0_fract
 - wiring.c, 46
- timer0_millis
 - wiring.c, 46
- timer0_overflow_count
 - wiring.c, 46
- timer1ISR
 - aDSEngine.cpp, 38
- toggleJoined
 - aDSChannels, 29
- toggleStandbyMode
 - aDSChannels, 29
- Tone.cpp, 45
- USBCore.cpp, 45
- updateLEDState
 - aDSChannel, 15
- updateOperationMode
 - aDSChannels, 29
- WInterrupts.c, 45
- WMath.cpp, 47
- WString.cpp, 47
- wiring.c, 45
 - delay, 46
 - delayMicroseconds, 46
 - f, 46
 - FRACT_INC, 46

FRACT_MAX, [46](#)
if, [46](#)
init, [46](#)
m, [46](#)
MILLIS_INC, [46](#)
micros, [46](#)
millis, [46](#)
timer0_fract, [46](#)
timer0_millis, [46](#)
timer0_overflow_count, [46](#)
wiring_analog.c, [46](#)
wiring_digital.c, [46](#)
wiring_pulse.c, [47](#)
wiring_shift.c, [47](#)