

Arduino

1.0

Generated by Doxygen 1.8.14



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	CpuLoad Class Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Constructor & Destructor Documentation . . . . .	5
3.1.2.1	CpuLoad() . . . . .	6
3.1.3	Member Function Documentation . . . . .	6
3.1.3.1	ComputeCPULoad() . . . . .	6
3.1.3.2	getAverageCPULoad() . . . . .	7
3.1.3.3	getCurrentCPULoad() . . . . .	7
3.1.3.4	getMaxCPULoad() . . . . .	8
3.2	dht22 Class Reference . . . . .	8
3.2.1	Detailed Description . . . . .	8
3.2.2	Constructor & Destructor Documentation . . . . .	9
3.2.2.1	dht22() . . . . .	9
3.2.3	Member Function Documentation . . . . .	9
3.2.3.1	read() . . . . .	9
3.3	dio Class Reference . . . . .	10
3.3.1	Detailed Description . . . . .	11

3.3.2	Constructor & Destructor Documentation . . . . .	11
3.3.2.1	dio() . . . . .	11
3.3.3	Member Function Documentation . . . . .	11
3.3.3.1	dio_changePortPinCnf() . . . . .	11
3.3.3.2	dio_getPort() . . . . .	12
3.3.3.3	dio_getPort_fast() . . . . .	12
3.3.3.4	dio_invertPort() . . . . .	13
3.3.3.5	dio_memorizePINaddress() . . . . .	14
3.3.3.6	dio_setPort() . . . . .	14
3.4	keepAliveLed Class Reference . . . . .	15
3.4.1	Detailed Description . . . . .	15
3.4.2	Constructor & Destructor Documentation . . . . .	15
3.4.2.1	keepAliveLed() . . . . .	16
3.4.3	Member Function Documentation . . . . .	16
3.4.3.1	blinkLed_task() . . . . .	16
3.5	scheduler Class Reference . . . . .	17
3.5.1	Detailed Description . . . . .	17
3.5.2	Constructor & Destructor Documentation . . . . .	18
3.5.2.1	scheduler() . . . . .	18
3.5.3	Member Function Documentation . . . . .	18
3.5.3.1	addPeriodicTask() . . . . .	18
3.5.3.2	getPitNumber() . . . . .	19
3.5.3.3	launchPeriodicTasks() . . . . .	20
3.5.3.4	removePeriodicTask() . . . . .	20
3.5.3.5	startScheduling() . . . . .	21
3.6	T_ASW_cnf_struct Struct Reference . . . . .	22
3.6.1	Detailed Description . . . . .	22
3.6.2	Member Data Documentation . . . . .	22
3.6.2.1	p_keepAliveLed . . . . .	22
3.6.2.2	p_TempSensor . . . . .	23

3.6.2.3	<a href="#">p_usartDebug</a>	23
3.7	<a href="#">T_BSW_cnf_struct Struct Reference</a>	23
3.7.1	<a href="#">Detailed Description</a>	24
3.7.2	<a href="#">Member Data Documentation</a>	24
3.7.2.1	<a href="#">p_cpuload</a>	24
3.7.2.2	<a href="#">p_dht22</a>	24
3.7.2.3	<a href="#">p_dio</a>	24
3.7.2.4	<a href="#">p_timer</a>	24
3.7.2.5	<a href="#">p_usart</a>	25
3.8	<a href="#">TempSensor Class Reference</a>	25
3.8.1	<a href="#">Detailed Description</a>	25
3.8.2	<a href="#">Constructor &amp; Destructor Documentation</a>	26
3.8.2.1	<a href="#">TempSensor()</a>	26
3.8.3	<a href="#">Member Function Documentation</a>	26
3.8.3.1	<a href="#">getHumidity()</a>	26
3.8.3.2	<a href="#">getHumPtr()</a>	27
3.8.3.3	<a href="#">getTemp()</a>	27
3.8.3.4	<a href="#">getTempPtr()</a>	28
3.8.3.5	<a href="#">readTempSensor_task()</a>	29
3.8.3.6	<a href="#">setValidity()</a>	29
3.8.3.7	<a href="#">updateLastValidValues()</a>	30
3.9	<a href="#">timer Class Reference</a>	31
3.9.1	<a href="#">Detailed Description</a>	31
3.9.2	<a href="#">Constructor &amp; Destructor Documentation</a>	31
3.9.2.1	<a href="#">timer()</a>	31
3.9.3	<a href="#">Member Function Documentation</a>	32
3.9.3.1	<a href="#">configureTimer1()</a>	32
3.9.3.2	<a href="#">getTimer1Value()</a>	33
3.9.3.3	<a href="#">startTimer1()</a>	34
3.9.3.4	<a href="#">stopTimer1()</a>	34

3.10	usart Class Reference . . . . .	34
3.10.1	Detailed Description . . . . .	35
3.10.2	Constructor & Destructor Documentation . . . . .	35
3.10.2.1	usart() . . . . .	35
3.10.3	Member Function Documentation . . . . .	36
3.10.3.1	setBaudRate() . . . . .	36
3.10.3.2	usart_init() . . . . .	36
3.10.3.3	usart_read() . . . . .	37
3.10.3.4	usart_sendString() . . . . .	37
3.11	UsartDebug Class Reference . . . . .	38
3.11.1	Detailed Description . . . . .	39
3.11.2	Constructor & Destructor Documentation . . . . .	39
3.11.2.1	UsartDebug() . . . . .	39
3.11.3	Member Function Documentation . . . . .	39
3.11.3.1	activateDebugMode() . . . . .	39
3.11.3.2	DebugModeManagement() . . . . .	40
3.11.3.3	DisplayCPULoad_task() . . . . .	41
3.11.3.4	DisplaySensors_task() . . . . .	42
3.11.3.5	isDebugModeActive() . . . . .	43
3.11.3.6	sendBool() . . . . .	43
3.11.3.7	sendInteger() . . . . .	44

<b>4</b>	<b>File Documentation</b>	<b>47</b>
4.1	work/asw/asw.cpp File Reference	47
4.1.1	Detailed Description	48
4.1.2	Function Documentation	48
4.1.2.1	asw_init()	48
4.1.3	Variable Documentation	48
4.1.3.1	ASW_cnf_struct	49
4.2	work/asw/asw.h File Reference	49
4.2.1	Detailed Description	50
4.2.2	Function Documentation	50
4.2.2.1	asw_init()	50
4.2.3	Variable Documentation	51
4.2.3.1	ASW_cnf_struct	51
4.3	work/asw/debug/debug.cpp File Reference	51
4.3.1	Detailed Description	52
4.3.2	Variable Documentation	52
4.3.2.1	str_debug_main_menu	52
4.4	work/asw/debug/debug.h File Reference	52
4.4.1	Detailed Description	54
4.4.2	Macro Definition Documentation	54
4.4.2.1	PERIOD_MS_TASK_DISPLAY_CPU_LOAD	54
4.4.2.2	PERIOD_MS_TASK_DISPLAY_SENSORS	54
4.4.3	Enumeration Type Documentation	54
4.4.3.1	debug_state_t	54
4.5	work/asw/keepAliveLed/keepAliveLed.cpp File Reference	55
4.5.1	Detailed Description	55
4.6	work/asw/keepAliveLed/keepAliveLed.h File Reference	56
4.6.1	Detailed Description	57
4.6.2	Macro Definition Documentation	57
4.6.2.1	LED_PORT	57

4.6.2.2	PERIOD_MS_TASK_LED . . . . .	57
4.7	work/asw/TempSensor/TempSensor.cpp File Reference . . . . .	58
4.7.1	Detailed Description . . . . .	58
4.7.2	Macro Definition Documentation . . . . .	58
4.7.2.1	PIT_BEFORE_INVALID . . . . .	59
4.8	work/asw/TempSensor/TempSensor.h File Reference . . . . .	59
4.8.1	Detailed Description . . . . .	60
4.8.2	Macro Definition Documentation . . . . .	60
4.8.2.1	PERIOD_MS_TASK_TEMP_SENSOR . . . . .	61
4.9	work/bsw/bsw.cpp File Reference . . . . .	61
4.9.1	Detailed Description . . . . .	62
4.9.2	Function Documentation . . . . .	62
4.9.2.1	bsw_init() . . . . .	62
4.9.3	Variable Documentation . . . . .	62
4.9.3.1	BSW_cnf_struct . . . . .	63
4.10	work/bsw/bsw.h File Reference . . . . .	63
4.10.1	Detailed Description . . . . .	64
4.10.2	Macro Definition Documentation . . . . .	64
4.10.2.1	USART_BAUDRATE . . . . .	64
4.10.3	Function Documentation . . . . .	64
4.10.3.1	bsw_init() . . . . .	65
4.10.4	Variable Documentation . . . . .	65
4.10.4.1	BSW_cnf_struct . . . . .	65
4.11	work/bsw/cpuLoad/CpuLoad.cpp File Reference . . . . .	66
4.11.1	Detailed Description . . . . .	66
4.12	work/bsw/cpuLoad/CpuLoad.h File Reference . . . . .	66
4.12.1	Detailed Description . . . . .	68
4.12.2	Macro Definition Documentation . . . . .	68
4.12.2.1	NB_OF_SAMPLES . . . . .	68
4.13	work/bsw/dht22/dht22.cpp File Reference . . . . .	68



4.13.1 Detailed Description . . . . .	69
4.13.2 Macro Definition Documentation . . . . .	69
4.13.2.1 MAX_WAIT_TIME_US . . . . .	69
4.14 work/bsw/dht22/dht22.h File Reference . . . . .	69
4.14.1 Detailed Description . . . . .	70
4.14.2 Macro Definition Documentation . . . . .	70
4.14.2.1 DHT22_PORT . . . . .	70
4.15 work/bsw/dio/dio.cpp File Reference . . . . .	71
4.15.1 Detailed Description . . . . .	71
4.16 work/bsw/dio/dio.h File Reference . . . . .	71
4.16.1 Detailed Description . . . . .	73
4.16.2 Macro Definition Documentation . . . . .	73
4.16.2.1 DECODE_PIN . . . . .	73
4.16.2.2 DECODE_PORT . . . . .	73
4.16.2.3 ENCODE_PORT . . . . .	73
4.16.2.4 PORT_A . . . . .	74
4.16.2.5 PORT_B . . . . .	74
4.16.2.6 PORT_C . . . . .	74
4.16.2.7 PORT_CNF_IN . . . . .	74
4.16.2.8 PORT_CNF_OUT . . . . .	74
4.16.2.9 PORT_D . . . . .	75
4.17 work/bsw/dio/dio_port_cnf.h File Reference . . . . .	75
4.17.1 Detailed Description . . . . .	75
4.17.2 Macro Definition Documentation . . . . .	76
4.17.2.1 PORTB_CNF_DDRB . . . . .	76
4.17.2.2 PORTB_CNF_PORTB . . . . .	76
4.18 work/bsw/dio/dio_reg_atm2560.h File Reference . . . . .	77
4.18.1 Macro Definition Documentation . . . . .	77
4.18.1.1 DDRA_PTR . . . . .	77
4.18.1.2 DDRB_PTR . . . . .	78

4.18.1.3	DDRC_PTR	78
4.18.1.4	DDRD_PTR	78
4.18.1.5	PINA_PTR	78
4.18.1.6	PINB_PTR	78
4.18.1.7	PINC_PTR	79
4.18.1.8	PIND_PTR	79
4.18.1.9	PORTA_PTR	79
4.18.1.10	PORTB_PTR	79
4.18.1.11	PORTC_PTR	79
4.18.1.12	PORTD_PTR	80
4.19	work/bsw/timer/timer.cpp File Reference	80
4.19.1	Detailed Description	80
4.20	work/bsw/timer/timer.h File Reference	81
4.20.1	Detailed Description	81
4.21	work/bsw/usart/usart.cpp File Reference	81
4.21.1	Detailed Description	82
4.22	work/bsw/usart/usart.h File Reference	82
4.22.1	Detailed Description	83
4.23	work/lib/operators.cpp File Reference	83
4.23.1	Detailed Description	84
4.23.2	Function Documentation	84
4.23.2.1	operator delete()	84
4.23.2.2	operator new()	84
4.24	work/lib/operators.h File Reference	85
4.24.1	Detailed Description	85
4.24.2	Function Documentation	86
4.24.2.1	operator delete()	86
4.24.2.2	operator new()	86
4.25	work/main.cpp File Reference	86
4.25.1	Detailed Description	87

4.25.2	Function Documentation	87
4.25.2.1	ISR() [1/2]	88
4.25.2.2	ISR() [2/2]	88
4.25.2.3	main()	89
4.26	work/main.h File Reference	89
4.26.1	Detailed Description	90
4.27	work/scheduler/scheduler.cpp File Reference	90
4.27.1	Detailed Description	91
4.27.2	Variable Documentation	91
4.27.2.1	p_scheduler	91
4.28	work/scheduler/scheduler.h File Reference	92
4.28.1	Detailed Description	93
4.28.2	Macro Definition Documentation	93
4.28.2.1	PRESCALER_PERIODIC_TIMER	93
4.28.2.2	SW_PERIOD_MS	93
4.28.2.3	TIMER_CTC_VALUE	94
4.28.3	Typedef Documentation	94
4.28.3.1	TaskPtr_t	94
4.28.4	Variable Documentation	94
4.28.4.1	p_scheduler	94
<b>Index</b>		<b>95</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">CpuLoad</a>	Class defining CPU load libraries . . . . .	5
<a href="#">dht22</a>	DHT 22 driver class . . . . .	8
<a href="#">dio</a>	DIO class . . . . .	10
<a href="#">keepAliveLed</a>	Class for keep-alive LED blinking . . . . .	15
<a href="#">scheduler</a>	Scheduler class . . . . .	17
<a href="#">T_ASW_cnf_struct</a>	ASW configuration structure . . . . .	22
<a href="#">T_BSW_cnf_struct</a>	BSW configuration structure . . . . .	23
<a href="#">TempSensor</a>	Class for temperature sensor . . . . .	25
<a href="#">timer</a>	Class defining a timer . . . . .	31
<a href="#">usart</a>	USART serial bus class . . . . .	34
<a href="#">UsartDebug</a>	Class used for debugging on usart link . . . . .	38



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

work/main.cpp	
Background task file	86
work/main.h	
Background task header file	89
work/asw/asw.cpp	
ASW main file	47
work/asw/asw.h	
ASW main header file	49
work/asw/debug/debug.cpp	
This file defines classes for log and debug data transmission on USART link	51
work/asw/debug/debug.h	
Header file for debug and logging functions	52
work/asw/keepAliveLed/keepAliveLed.cpp	
Definition of function for class <code>keepAliveLed</code>	55
work/asw/keepAliveLed/keepAliveLed.h	
Class <code>keepAliveLed</code> header file	56
work/asw/TempSensor/TempSensor.cpp	
Defines function of class <code>TempSensor</code>	58
work/asw/TempSensor/TempSensor.h	
Class <code>TempSensor</code> header file	59
work/bsw/bsw.cpp	
BSW main file	61
work/bsw/bsw.h	
BSW main header file	63
work/bsw/cpuLoad/CpuLoad.cpp	
Defines functions of class <code>CpuLoad</code>	66
work/bsw/cpuLoad/CpuLoad.h	
Class <code>CpuLoad</code> header file	66
work/bsw/dht22/dht22.cpp	
This file defines classes for DHT22 driver	68
work/bsw/dht22/dht22.h	
DHT22 driver header file	69
work/bsw/dio/dio.cpp	
DIO library	71
work/bsw/dio/dio.h	
DIO library header file	71

work/bsw/dio/ <a href="#">dio_port_cnf.h</a>	
Digital ports configuration file . . . . .	75
work/bsw/dio/ <a href="#">dio_reg_atm2560.h</a> . . . . .	77
work/bsw/timer/ <a href="#">timer.cpp</a>	
Defines function for class timer . . . . .	80
work/bsw/timer/ <a href="#">timer.h</a>	
Timer class header file . . . . .	81
work/bsw/usart/ <a href="#">usart.cpp</a>	
BSW library for USART . . . . .	81
work/bsw/usart/ <a href="#">usart.h</a>	
Header file for USART library . . . . .	82
work/lib/ <a href="#">operators.cpp</a>	
C++ operators definitions . . . . .	83
work/lib/ <a href="#">operators.h</a>	
C++ operators definitions header file . . . . .	85
work/scheduler/ <a href="#">scheduler.cpp</a>	
Defines scheduler class . . . . .	90
work/scheduler/ <a href="#">scheduler.h</a>	
Scheduler class header file . . . . .	92



## Chapter 3

# Class Documentation

### 3.1 CpuLoad Class Reference

Class defining CPU load libraries.

```
#include <CpuLoad.h>
```

#### Public Member Functions

- [CpuLoad \(\)](#)  
*CpuLoad class constructor.*
- void [ComputeCPULoad \(\)](#)  
*Computes current CPU load.*
- uint8\_t [getCurrentCPULoad \(\)](#)  
*Get current CPU load value.*
- uint8\_t [getAverageCPULoad \(\)](#)  
*Get average CPU load value.*
- uint8\_t [getMaxCPULoad \(\)](#)  
*Get maximum CPU load value.*

#### 3.1.1 Detailed Description

Class defining CPU load libraries.

This class defines tools to compute and monitor CPU load.

Definition at line 19 of file CpuLoad.h.

#### 3.1.2 Constructor & Destructor Documentation

### 3.1.2.1 CpuLoad()

```
CpuLoad::CpuLoad ( )
```

[CpuLoad](#) class constructor.

This function initializes class [CpuLoad](#)

#### Returns

Nothing

Definition at line 13 of file CpuLoad.cpp.

## 3.1.3 Member Function Documentation

### 3.1.3.1 ComputeCPULoad()

```
void CpuLoad::ComputeCPULoad ( )
```

Computes current CPU load.

This function computes the current CPU load using value of the timer used by the scheduler at the end of the periodic cycle. This value is divided by the PIT period to obtain CPU load;

#### Returns

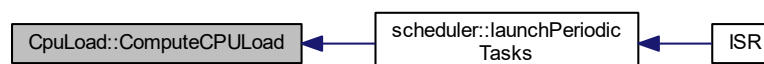
Nothing

Definition at line 27 of file CpuLoad.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 3.1.3.2 getAverageCPULoad()

```
uint8_t CpuLoad::getAverageCPULoad ( ) [inline]
```

Get average CPU load value.

This function returns the average CPU load value

#### Returns

Average CPU load value

Definition at line 56 of file CpuLoad.h.

Here is the caller graph for this function:



### 3.1.3.3 getCurrrrentCPULoad()

```
uint8_t CpuLoad::getCurrrrentCPULoad ( ) [inline]
```

Get current CPU load value.

This function returns the current CPU load value

#### Returns

Current CPU load value

Definition at line 45 of file CpuLoad.h.

Here is the caller graph for this function:



### 3.1.3.4 getMaxCPULoad()

```
uint8_t CpuLoad::getMaxCPULoad ( ) [inline]
```

Get maximum CPU load value.

This function returns the maximum CPU load value

#### Returns

Maximum CPU load value

Definition at line 67 of file CpuLoad.h.

Here is the caller graph for this function:



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

- [work/bsw/cpuLoad/CpuLoad.h](#)
- [work/bsw/cpuLoad/CpuLoad.cpp](#)

## 3.2 dht22 Class Reference

DHT 22 driver class.

```
#include <dht22.h>
```

### Public Member Functions

- [dht22](#) ()  
*dht22 class constructor*
- bool [read](#) (uint16\_t \*raw\_humidity, uint16\_t \*raw\_temperature)  
*Reads the data from DHT22.*

### 3.2.1 Detailed Description

DHT 22 driver class.

This class defines all useful functions for DHT22 temperature and humidity sensor

Definition at line 22 of file dht22.h.

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 dht22()

```
dht22::dht22 ( )
```

[dht22](#) class constructor

Initializes the class [dht22](#)

##### Returns

Nothing

Definition at line 22 of file dht22.cpp.

### 3.2.3 Member Function Documentation

#### 3.2.3.1 read()

```
bool dht22::read (
    uint16_t * raw_humidity,
    uint16_t * raw_temperature )
```

Reads the data from DHT22.

This function communicates with DHT22 using 1-wire protocol to read raw values of temperature and humidity. A checksum check is done when communication is finished to validate the received data

##### Parameters

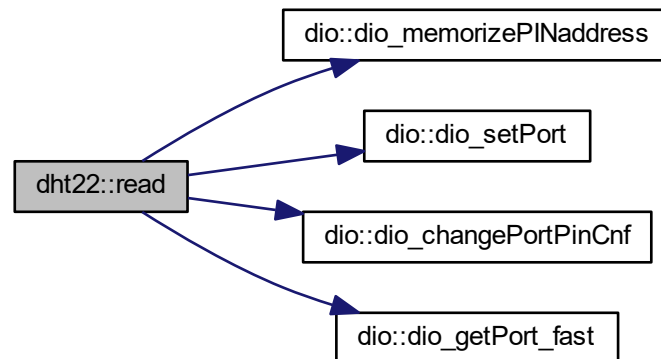
out	<i>raw_humidity</i>	Raw humidity value received from sensor
out	<i>raw_temperature</i>	Raw temperature value received from sensor

##### Returns

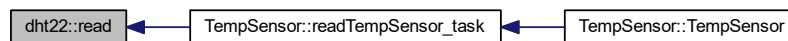
Validity of the read value

Definition at line 27 of file dht22.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- [work/bsw/dht22/dht22.h](#)
- [work/bsw/dht22/dht22.cpp](#)

### 3.3 dio Class Reference

DIO class.

```
#include <dio.h>
```

#### Public Member Functions

- [dio](#) ()  
*dio class constructor*
- void [dio\\_setPort](#) (uint8\_t portcode, bool state)  
*Port setting function.*
- void [dio\\_invertPort](#) (uint8\_t portcode)  
*Inverts the state of output port.*

- bool [dio\\_getPort](#) (uint8\_t portcode)  
*Gets the logical state of selected pin.*
- bool [dio\\_getPort\\_fast](#) (void)  
*Gets the logical state of the memorized pin.*
- void [dio\\_changePortPinCnf](#) (uint8\_t portcode, uint8\_t cnf)  
*Changes the IO configuration of the selected pin.*
- void [dio\\_memorizePINaddress](#) (uint8\_t portcode)  
*Memorizes PINx register address and pin index.*

### 3.3.1 Detailed Description

DIO class.

This class defines all useful functions for digital input/output ports

Definition at line 31 of file dio.h.

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 dio()

```
dio::dio ( )
```

dio class constructor

Initializes class dio and calls DIO hardware initialization function

#### Returns

Nothing

Definition at line 112 of file dio.cpp.

### 3.3.3 Member Function Documentation

#### 3.3.3.1 dio\_changePortPinCnf()

```
void dio::dio_changePortPinCnf (
    uint8_t portcode,
    uint8_t cnf )
```

Changes the IO configuration of the selected pin.

This function configures the selected pin as input or output according to parameter cnf. The corresponding port and pin index is extracted from parameter portcode.

**Parameters**

in	<i>portcode</i>	Encoded pin and register index
in	<i>cnf</i>	Requested configuration for the selected pin PORT_CNF_OUT (1) : pin configured as output PORT_CNF_IN (0) : pin configured as input

**Returns**

Nothing

Definition at line 149 of file dio.cpp.

Here is the caller graph for this function:

**3.3.3.2 dio\_getPort()**

```
bool dio::dio_getPort (
    uint8_t portcode )
```

Gets the logical state of selected pin.

This function gets the logical value of the selected pin. The corresponding port and pin index is extracted from parameter portcode.

**Parameters**

in	<i>portcode</i>	Encoded pin and register index
----	-----------------	--------------------------------

**Returns**

Logical state of selected pin

Definition at line 139 of file dio.cpp.

**3.3.3.3 dio\_getPort\_fast()**

```
bool dio::dio_getPort_fast (
    void )
```



Gets the logical state of the memorized pin.

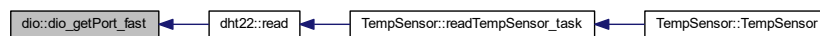
This function gets the logical value of the memorized pin. The corresponding port and pin index are stored in class members `PINx_addr_mem` and `PINx_idx_mem`. This mechanism is used to speed up reading time as this function no longer needs to extract register address and pin index from portcode.

#### Returns

Logical state of selected pin

Definition at line 171 of file `dio.cpp`.

Here is the caller graph for this function:



#### 3.3.3.4 dio\_invertPort()

```
void dio::dio_invertPort (
    uint8_t portcode )
```

Inverts the state of output port.

This function inverts the state of the chosen pin. The corresponding port and pin index is extracted from parameter `portcode`.

#### Parameters

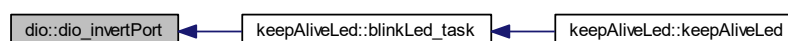
in	<i>portcode</i>	Encoded pin and register index
----	-----------------	--------------------------------

#### Returns

Nothing

Definition at line 131 of file `dio.cpp`.

Here is the caller graph for this function:



### 3.3.3.5 dio\_memorizePINaddress()

```
void dio::dio_memorizePINaddress (
    uint8_t portcode )
```

Memorizes PINx register address and pin index.

This function is used to speed up reading of register PINx. Register address and pin index are decoded from portcode parameter and stored for later use by function dio\_getPort\_fast.

#### Parameters

in	<i>portcode</i>	Encoded pin and register index
----	-----------------	--------------------------------

#### Returns

Nothing

Definition at line 165 of file dio.cpp.

Here is the caller graph for this function:



### 3.3.3.6 dio\_setPort()

```
void dio::dio_setPort (
    uint8_t portcode,
    bool state )
```

Port setting function.

This function sets the requested digital output to the requested state. The corresponding port and pin index is extracted from parameter portcode.

#### Parameters

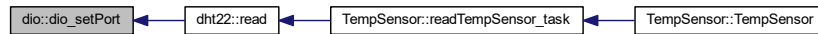
in	<i>portcode</i>	Encoded pin and register index
in	<i>state</i>	Requested state to set pin

#### Returns

Nothing

Definition at line 121 of file dio.cpp.

Here is the caller graph for this function:



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

- [work/bsw/dio/dio.h](#)
- [work/bsw/dio/dio.cpp](#)

## 3.4 keepAliveLed Class Reference

Class for keep-alive LED blinking.

```
#include <keepAliveLed.h>
```

### Public Member Functions

- [keepAliveLed \(\)](#)  
*Class constructor.*

### Static Public Member Functions

- static void [blinkLed\\_task \(\)](#)  
*Task for LED blinking.*

#### 3.4.1 Detailed Description

Class for keep-alive LED blinking.

This class defines all functions to make keep-alive LED blink

Definition at line 22 of file keepAliveLed.h.

#### 3.4.2 Constructor & Destructor Documentation

### 3.4.2.1 keepAliveLed()

```
keepAliveLed::keepAliveLed ( )
```

Class constructor.

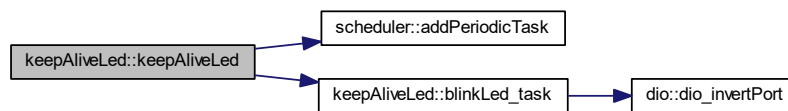
This function initializes the class keepAliveLed

#### Returns

Nothing

Definition at line 15 of file keepAliveLed.cpp.

Here is the call graph for this function:



## 3.4.3 Member Function Documentation

### 3.4.3.1 blinkLed\_task()

```
void keepAliveLed::blinkLed_task ( ) [static]
```

Task for LED blinking.

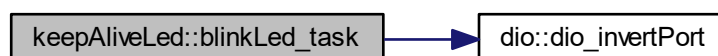
This function is inserted into the scheduler. It changes the state of the LED output to make it blink

#### Returns

Nothing

Definition at line 21 of file keepAliveLed.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- [work/asw/keepAliveLed/keepAliveLed.h](#)
- [work/asw/keepAliveLed/keepAliveLed.cpp](#)

## 3.5 scheduler Class Reference

Scheduler class.

```
#include <scheduler.h>
```

### Public Member Functions

- [scheduler \(\)](#)  
*scheduler class constructor*
- void [launchPeriodicTasks \(\)](#)  
*Main scheduler function.*
- void [startScheduling \(\)](#)  
*Starts the tasks scheduling.*
- void [addPeriodicTask \(TaskPtr\\_t task\\_ptr, uint16\\_t a\\_period\)](#)  
*Add a task into the scheduler.*
- bool [removePeriodicTask \(TaskPtr\\_t task\\_ptr\)](#)  
*Remove a task from the scheduler.*
- uint32\_t [getPitNumber \(\)](#)  
*Get function for PIT number.*

### 3.5.1 Detailed Description

Scheduler class.

This class defines the scheduler of the system. It is called by the main interrupt and calls successively all applicative functions according to their recurrence time.

Definition at line 32 of file `scheduler.h`.

### 3.5.2 Constructor & Destructor Documentation

#### 3.5.2.1 scheduler()

```
scheduler::scheduler ( )
```

scheduler class constructor

This function initializes the class scheduler

#### Returns

Nothing

Definition at line 19 of file scheduler.cpp.

Here is the call graph for this function:



### 3.5.3 Member Function Documentation

#### 3.5.3.1 addPeriodicTask()

```
void scheduler::addPeriodicTask (
    TaskPtr_t task_ptr,
    uint16_t a_period )
```

Add a task into the scheduler.

This function create a new task in the scheduler linked to the function task\_ptr with a period a\_period and an ID a\_task\_id

#### Parameters

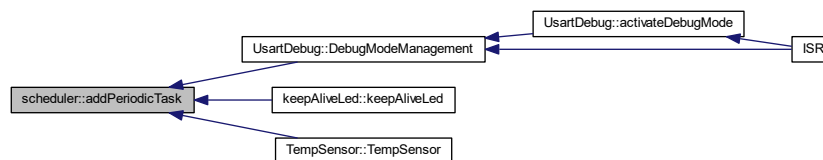
in	<i>task_ptr</i>	Pointer to the task which will be added
in	<i>a_period</i>	Period of the new task

**Returns**

Nothing

Definition at line 66 of file scheduler.cpp.

Here is the caller graph for this function:

**3.5.3.2 getPitNumber()**

```
uint32_t scheduler::getPitNumber ( )
```

Get function for PIT number.

This function returns the PIT number

**Returns**

PIT number

Definition at line 96 of file scheduler.cpp.

Here is the caller graph for this function:



### 3.5.3.3 launchPeriodicTasks()

```
void scheduler::launchPeriodicTasks ( )
```

Main scheduler function.

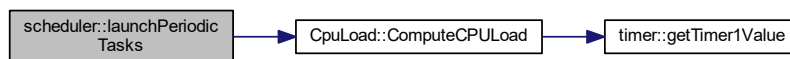
This function launches the scheduled tasks according to current software time and task configuration

#### Returns

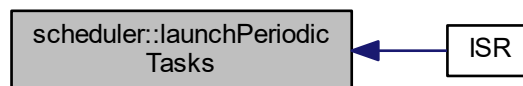
Nothing

Definition at line 32 of file scheduler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 3.5.3.4 removePeriodicTask()

```
bool scheduler::removePeriodicTask (
    TaskPtr_t task_ptr )
```

Remove a task from the scheduler.

This function finds the task defined by `task_ptr` in the scheduler and removes it.

#### Parameters

in	<i>task_ptr</i>	address of the task to remove from scheduler
----	-----------------	--



**Returns**

TRUE if the task has been removed, FALSE if the task does not exist in the scheduler

Definition at line 102 of file scheduler.cpp.

Here is the caller graph for this function:

**3.5.3.5 startScheduling()**

```
void scheduler::startScheduling ( )
```

Starts the tasks scheduling.

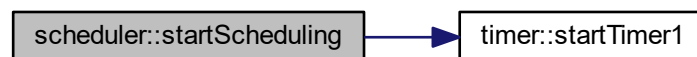
This function starts the timer which will trigger an interrupt every software period. When the interrupt is raised the scheduler will launch applications

**Returns**

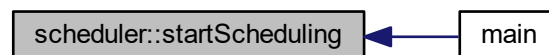
Nothing

Definition at line 60 of file scheduler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

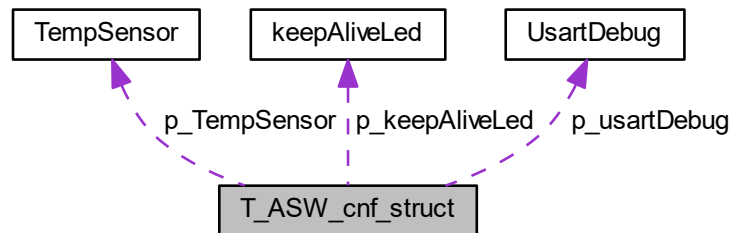
- [work/scheduler/scheduler.h](#)
- [work/scheduler/scheduler.cpp](#)

### 3.6 T\_ASW\_cnf\_struct Struct Reference

ASW configuration structure.

```
#include <asw.h>
```

Collaboration diagram for T\_ASW\_cnf\_struct:



#### Public Attributes

- [UsartDebug](#) \* [p\\_usartDebug](#)
- [keepAliveLed](#) \* [p\\_keepAliveLed](#)
- [TempSensor](#) \* [p\\_TempSensor](#)

#### 3.6.1 Detailed Description

ASW configuration structure.

This structure contains all pointers to instanced applicative objects

Definition at line 23 of file asw.h.

#### 3.6.2 Member Data Documentation

##### 3.6.2.1 p\_keepAliveLed

```
keepAliveLed* T_ASW_cnf_struct::p_keepAliveLed
```

Pointer to [keepAliveLed](#) object

Definition at line 26 of file asw.h.

### 3.6.2.2 p\_TempSensor

```
TempSensor* T_ASW_cnf_struct::p_TempSensor
```

Pointer to [TempSensor](#) object

Definition at line 27 of file asw.h.

### 3.6.2.3 p\_usartDebug

```
UsartDebug* T_ASW_cnf_struct::p_usartDebug
```

Pointer to usart debug object

Definition at line 25 of file asw.h.

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

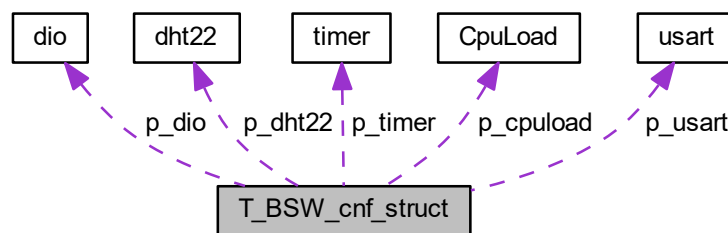
- [work/asw/asw.h](#)

## 3.7 T\_BSW\_cnf\_struct Struct Reference

BSW configuration structure.

```
#include <bsw.h>
```

Collaboration diagram for T\_BSW\_cnf\_struct:



### Public Attributes

- [usart](#) \* [p\\_usart](#)
- [dio](#) \* [p\\_dio](#)
- [timer](#) \* [p\\_timer](#)
- [dht22](#) \* [p\\_dht22](#)
- [CpuLoad](#) \* [p\\_cpuload](#)

### 3.7.1 Detailed Description

BSW configuration structure.

This structure contains all pointers to instanced drivers objects

Definition at line 30 of file bsw.h.

### 3.7.2 Member Data Documentation

#### 3.7.2.1 p\_cpuload

```
CpuLoad* T_BSW_cnf_struct::p_cpuload
```

Pointer to cpu load library object

Definition at line 36 of file bsw.h.

#### 3.7.2.2 p\_dht22

```
dht22* T_BSW_cnf_struct::p_dht22
```

Pointer to [dht22](#) driver object

Definition at line 35 of file bsw.h.

#### 3.7.2.3 p\_dio

```
dio* T_BSW_cnf_struct::p_dio
```

Pointer to dio driver object

Definition at line 33 of file bsw.h.

#### 3.7.2.4 p\_timer

```
timer* T_BSW_cnf_struct::p_timer
```

Pointer to timer driver object

Definition at line 34 of file bsw.h.

### 3.7.2.5 p\_usart

```
usart* T_BSW_cnf_struct::p_usart
```

Pointer to usart driver object

Definition at line 32 of file bsw.h.

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

- [work/bsw/bsw.h](#)

## 3.8 TempSensor Class Reference

Class for temperature sensor.

```
#include <TempSensor.h>
```

### Public Member Functions

- [TempSensor](#) ()  
*Class constructor.*
- uint16\_t \* [getTempPtr](#) ()  
*Get pointer to data raw\_temperature.*
- uint16\_t \* [getHumPtr](#) ()  
*Get pointer to data raw\_humidity.*
- bool [getTemp](#) (uint16\_t \*temp)  
*Get temperature data.*
- bool [getHumidity](#) (uint16\_t \*hum)  
*Get humidity data.*
- void [setValidity](#) (bool validity)  
*Set data val\_validity.*
- void [updateLastValidValues](#) ()

### Static Public Member Functions

- static void [readTempSensor\\_task](#) ()  
*Task for reading temperature and humidity values.*

### 3.8.1 Detailed Description

Class for temperature sensor.

This class defines all functions used to read data from temperature sensor and monitor it

Definition at line 19 of file TempSensor.h.

### 3.8.2 Constructor & Destructor Documentation

#### 3.8.2.1 TempSensor()

```
TempSensor::TempSensor ( )
```

Class constructor.

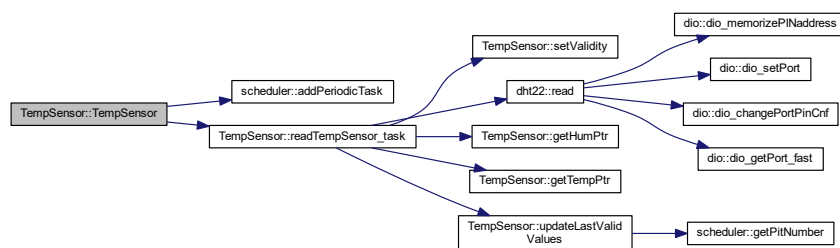
This function initializes all data of the class [TempSensor](#)

##### Returns

Nothing

Definition at line 16 of file TempSensor.cpp.

Here is the call graph for this function:



### 3.8.3 Member Function Documentation

#### 3.8.3.1 getHumidity()

```
bool TempSensor::getHumidity (
    uint16_t * hum )
```

Get humidity data.

This function returns the value of the humidity. If the official value is not valid, the function return false.

##### Parameters

out	hum	Humidity value
-----	-----	----------------

**Returns**

Validity of humidity

Definition at line 66 of file TempSensor.cpp.

Here is the caller graph for this function:

**3.8.3.2 getHumPtr()**

```
uint16_t * TempSensor::getHumPtr ( )
```

Get pointer to data raw\_humidity.

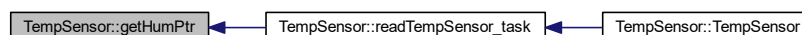
This function returns a pointer to the class member raw\_humidity

**Returns**

Pointer to raw\_humidity

Definition at line 41 of file TempSensor.cpp.

Here is the caller graph for this function:

**3.8.3.3 getTemp()**

```
bool TempSensor::getTemp (
    uint16_t * temp )
```

Get temperature data.

This function returns the value of the temperature. If the official value is not valid, the function return false.

**Parameters**

out	<i>temp</i>	Temperature value
-----	-------------	-------------------

**Returns**

Validity of temperature

Definition at line 72 of file TempSensor.cpp.

Here is the caller graph for this function:

**3.8.3.4 getTempPtr()**

```
uint16_t * TempSensor::getTempPtr ( )
```

Get pointer to data raw\_temperature.

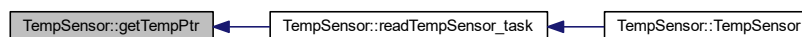
This function returns a pointer to the class member raw\_temperature

**Returns**

Pointer to raw\_temperature

Definition at line 46 of file TempSensor.cpp.

Here is the caller graph for this function:





## 3.8.3.5 readTempSensor\_task()

```
void TempSensor::readTempSensor_task ( ) [static]
```

Task for reading temperature and humidity values.

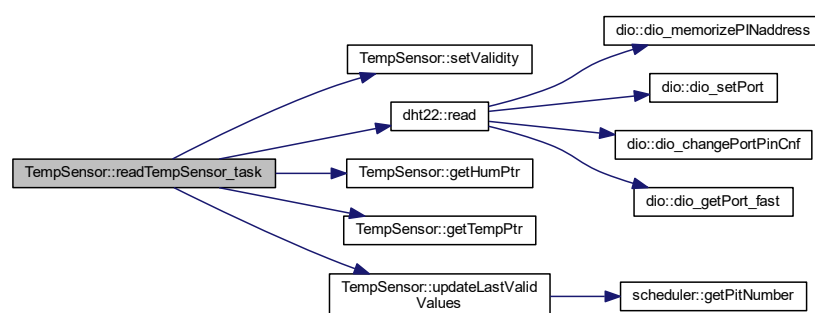
This task reads temperature and humidity data using DHT22 driver. It is called every 5 seconds.

## Returns

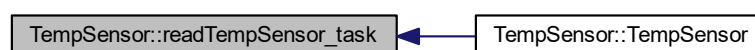
Nothing

Definition at line 30 of file TempSensor.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 3.8.3.6 setValidity()

```
void TempSensor::setValidity (
    bool validity )
```

Set data `val_validity`.

This function sets the class member `val_validity`

**Parameters**

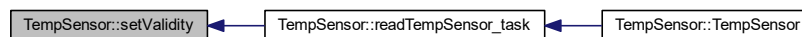
in	<i>validity</i>	Value of validity
----	-----------------	-------------------

**Returns**

Nothing

Definition at line 36 of file TempSensor.cpp.

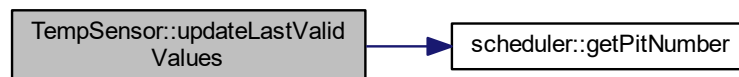
Here is the caller graph for this function:

**3.8.3.7 updateLastValidValues()**

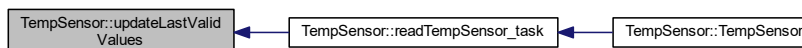
```
void TempSensor::updateLastValidValues ( )
```

Definition at line 51 of file TempSensor.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- [work/asw/TempSensor/TempSensor.h](#)
- [work/asw/TempSensor/TempSensor.cpp](#)

## 3.9 timer Class Reference

Class defining a timer.

```
#include <timer.h>
```

### Public Member Functions

- [timer](#) ()  
*Class constructor.*
- void [configureTimer1](#) (uint16\_t a\_prescaler, uint16\_t a\_ctcValue)  
*Configures Timer #1.*
- void [startTimer1](#) ()  
*Start Timer #1.*
- void [stopTimer1](#) ()  
*Stops Timer #1.*
- uint16\_t [getTimer1Value](#) ()  
*Reads current value of timer #1.*

### 3.9.1 Detailed Description

Class defining a timer.

This class defines a timer/counter. The selected timer is configured in CTC mode and interrupts are enabled. The prescaler value and CTC value can both be configured by user.

Definition at line 22 of file timer.h.

### 3.9.2 Constructor & Destructor Documentation

#### 3.9.2.1 timer()

```
timer::timer ( )
```

Class constructor.

This function initializes class attributes

#### Returns

Nothing

Definition at line 13 of file timer.cpp.

### 3.9.3 Member Function Documentation

#### 3.9.3.1 configureTimer1()

```
void timer::configureTimer1 (
    uint16_t a_prescaler,
    uint16_t a_ctcValue )
```

Configures Timer #1.

This function configures hardware timer #1 in CTC mode, enables its interrupts, sets prescaler to `a_prescaler` and CTC value to `a_ctcValue`

## Parameters

in	<i>a_prescaler</i>	prescaler value
in	<i>a_ctcValue</i>	Value to which the counter will compare before raising an interrupt

## Returns

Nothing

Definition at line 18 of file timer.cpp.

Here is the caller graph for this function:



### 3.9.3.2 getTimer1Value()

```
uint16_t timer::getTimer1Value ( ) [inline]
```

Reads current value of timer #1.

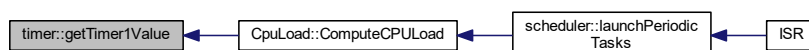
This function reads the value of of timer #1 using register TCNT1. The function is inlined to speed up SW execution.

## Returns

Current timer value

Definition at line 61 of file timer.h.

Here is the caller graph for this function:



### 3.9.3.3 startTimer1()

```
void timer::startTimer1 ( )
```

Start Timer #1.

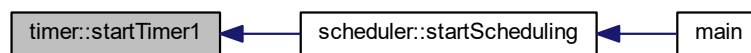
This functions starts Timer #1. Timer shall be initialized before this function is called.

#### Returns

Nothing

Definition at line 56 of file timer.cpp.

Here is the caller graph for this function:



### 3.9.3.4 stopTimer1()

```
void timer::stopTimer1 ( )
```

Stops Timer #1.

This functions stops timer #1 by resetting bits 0-2 of TCCR1B

#### Returns

Nothing

Definition at line 67 of file timer.cpp.

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

- [work/bsw/timer/timer.h](#)
- [work/bsw/timer/timer.cpp](#)

## 3.10 usart Class Reference

USART serial bus class.

```
#include <usart.h>
```

## Public Member Functions

- `usart` (`uint16_t a_BaudRate`)  
*Class usart constructor.*
- `void usart_sendString` (`uint8_t *str`)  
*Sending a string on USART link.*
- `void setBaudRate` (`uint16_t a_BaudRate`)  
*Setting baud rate.*
- `void usart_init` ()  
*USART hardware initialization.*
- `uint8_t usart_read` ()  
*USART read function.*

### 3.10.1 Detailed Description

USART serial bus class.

This class defines all useful functions for USART serial bus

Definition at line 16 of file usart.h.

### 3.10.2 Constructor & Destructor Documentation

#### 3.10.2.1 usart()

```
usart::usart (  
    uint16_t a_BaudRate )
```

Class usart constructor.

Initializes the class and call hardware initialization function

#### Parameters

in	<code>a_BaudRate</code>	Desired Baud Rate (16 bit) - up to 57600
----	-------------------------	--

#### Returns

Nothing.

Definition at line 14 of file usart.cpp.

Here is the call graph for this function:



### 3.10.3 Member Function Documentation

#### 3.10.3.1 setBaudRate()

```
void uart::setBaudRate (
    uint16_t a_BaudRate ) [inline]
```

Setting baud rate.

This function sets the attribute BaudRate of the class uart

##### Parameters

in	<i>a_BaudRate</i>	Desired Baud Rate (16 bit) - up to 57600
----	-------------------	--

##### Returns

Nothing

Definition at line 63 of file uart.cpp.

#### 3.10.3.2 uart\_init()

```
void uart::uart_init ( )
```

USART hardware initialization.

This function will initialize the USART using selected baudrate. User must pay attention to select one of the usually used Baud Rate (9600, 19200, 38400, 57600). Note that since an uint16 is used as argument, Baud rate cannot be more than 57600.



**Returns**

Nothing.

Definition at line 21 of file usart.cpp.

Here is the caller graph for this function:

**3.10.3.3 usart\_read()**

```
uint8_t usart::usart_read ( )
```

USART read function.

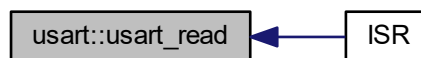
This function will read reception register of USART

**Returns**

The function returns the 8 bits read from reception buffer

Definition at line 79 of file usart.cpp.

Here is the caller graph for this function:

**3.10.3.4 usart\_sendString()**

```
void usart::usart_sendString (
    uint8_t * str )
```

Sending a string on USART link.

Just write data to the Serial link using `usart_trabsmit` function

## Parameters

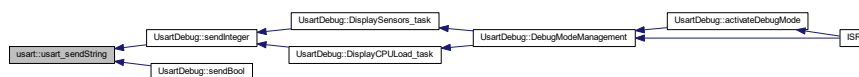
in	str	Pointer to the string being sent
----	-----	----------------------------------

## Returns

Nothing.

Definition at line 44 of file usart.cpp.

Here is the caller graph for this function:



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

- work/bsw/usart/[usart.h](#)
- work/bsw/usart/[usart.cpp](#)

### 3.11 UsartDebug Class Reference

Class used for debugging on usart link.

```
#include <debug.h>
```

#### Public Member Functions

- [UsartDebug](#) ()  
*Class [UsartDebug](#) constructor.*
- void [sendInteger](#) (uint16\_t data, uint8\_t base)  
*Send a integer data on USART link.*
- void [sendBool](#) (bool data)  
*Send a boolean data on USART link.*
- bool [isDebugModeActive](#) ()  
*Check is debug mode is active or not.*
- void [activateDebugMode](#) ()  
*Activates debug mode.*
- void [DebugModeManagement](#) (uint8\_t rcv\_char)  
*Management of debug mode.*

#### Static Public Member Functions

- static void [DisplaySensors\\_task](#) ()  
*Displays sensors data on usart link.*
- static void [DisplayCPULoad\\_task](#) ()  
*Displays CPU load data on usart link.*

### 3.11.1 Detailed Description

Class used for debugging on usart link.

This class defines functions used for sending debug data on USART link.

Definition at line 33 of file debug.h.

### 3.11.2 Constructor & Destructor Documentation

#### 3.11.2.1 UsartDebug()

```
UsartDebug::UsartDebug ( )
```

Class [UsartDebug](#) constructor.

Initializes the class [UsartDebug](#)

#### Returns

Nothing

Definition at line 31 of file debug.cpp.

### 3.11.3 Member Function Documentation

#### 3.11.3.1 activateDebugMode()

```
void UsartDebug::activateDebugMode ( )
```

Activates debug mode.

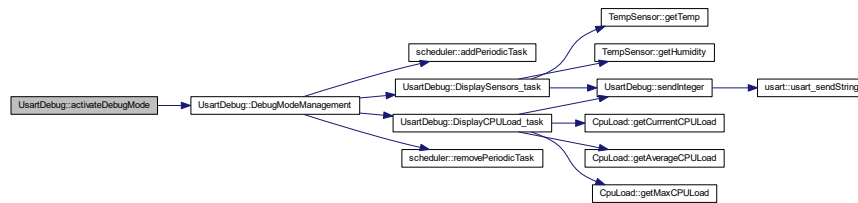
This function activates USART debug mode.

**Returns**

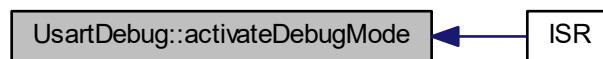
Nothing

Definition at line 126 of file debug.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**3.11.3.2 DebugModeManagement()**

```
void UsartDebug::DebugModeManagement (
    uint8_t rcv_char )
```

Management of debug mode.

This function manages the debug mode according to the following state machine :

- init state : display main menu
- WAIT\_INIT state : handles user choice in main menu and selects next state
- DISPLAY\_DATA state : display sensor data periodically
- DISPLAY CPU LOAD : display CPU load periodically

It is called each time a data is received on USART and debug mode is active

## Parameters

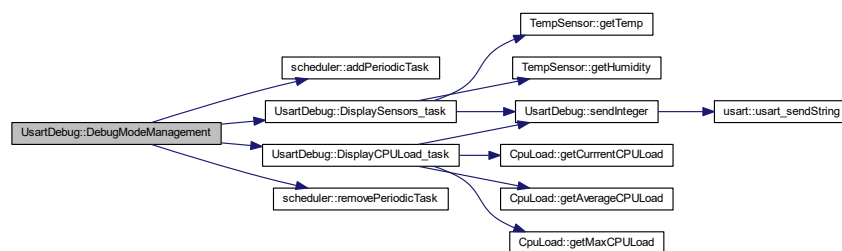
in	<i>rcv_char</i>	8 bits character received on USART
----	-----------------	------------------------------------

## Returns

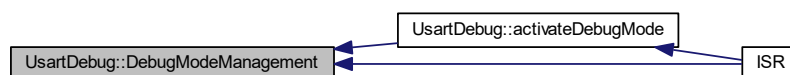
Nothing

Definition at line 134 of file debug.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 3.11.3.3 DisplayCPULoad\_task()

```
void UsartDebug::DisplayCPULoad_task ( ) [static]
```

Displays CPU load data on usart link.

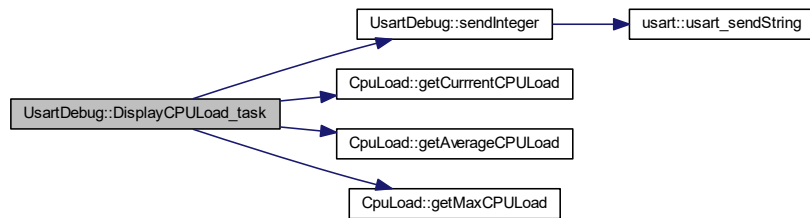
This task sends CPU load data (current and average load) on usart link every 5 seconds

**Returns**

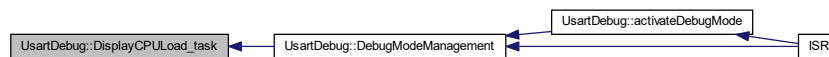
Nothing

Definition at line 110 of file debug.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**3.11.3.4 DisplaySensors\_task()**

```
void UsartDebug::DisplaySensors_task ( ) [static]
```

Displays sensors data on usart link.

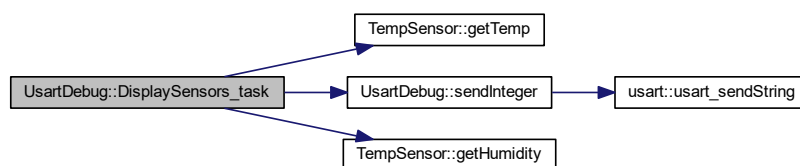
This task sends sensors data (temperature and humidity) on usart link every 5 seconds

**Returns**

Nothing

Definition at line 69 of file debug.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 3.11.3.5 isDebugModeActive()

```
bool UsartDebug::isDebugModeActive ( )
```

Check is debug mode is active or not.

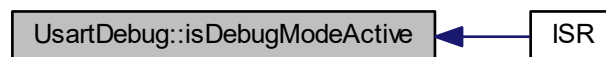
This function checks if debug mode is active or not.

#### Returns

TRUE is debug mode is active, FALSE otherwise

Definition at line 121 of file debug.cpp.

Here is the caller graph for this function:



### 3.11.3.6 sendBool()

```
void UsartDebug::sendBool (
    bool data )
```

Send a boolean data on USART link.

This functions sends the requested boolean on USART link by calling driver's transmission function. The boolean data is first converted into a string and then sent

**Parameters**

in	<i>data</i>	boolean data to be sent
----	-------------	-------------------------

**Returns**

Nothing

Definition at line 57 of file debug.cpp.

Here is the call graph for this function:

**3.11.3.7 sendInteger()**

```
void UsartDebug::sendInteger (
    uint16_t data,
    uint8_t base )
```

Send a integer data on USART link.

This functions sends the requested integer on USART link by calling driver's transmission function. The integer is first converted into a string and then sent

**Parameters**

in	<i>data</i>	integer data to be sent
in	<i>base</i>	numerical base used to convert integer into string (between 2 and 36)

**Returns**

Nothing

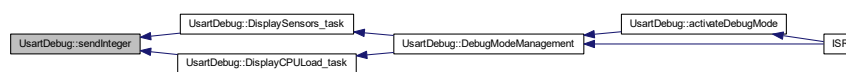
Definition at line 43 of file debug.cpp.



Here is the call graph for this function:



Here is the caller graph for this function:



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

- [work/asw/debug/debug.h](#)
- [work/asw/debug/debug.cpp](#)



## Chapter 4

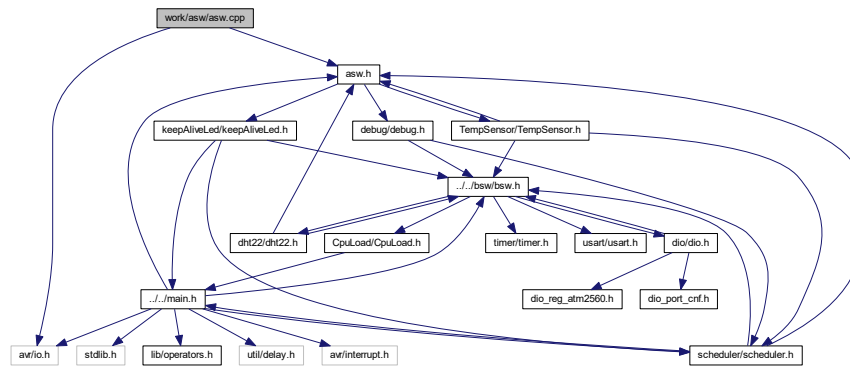
# File Documentation

### 4.1 work/asw/asw.cpp File Reference

ASW main file.

```
#include <avr/io.h>
#include "asw.h"
```

Include dependency graph for asw.cpp:



### Functions

- void `asw_init()`  
*Initialization of ASW.*

### Variables

- `T_ASW_cnf_struct` `ASW_cnf_struct`

### 4.1.1 Detailed Description

ASW main file.

#### Date

15 mars 2018

#### Author

nicls67

### 4.1.2 Function Documentation

#### 4.1.2.1 asw\_init()

```
void asw_init ( )
```

Initialization of ASW.

This function instantiates all applicative objects. The addresses of objects are then stored in ASW\_cnf\_struct structure. This function shall be called after BSW initialization function.

#### Returns

Nothing

Definition at line 20 of file asw.cpp.

Here is the caller graph for this function:



### 4.1.3 Variable Documentation

## 4.1.3.1 ASW\_cnf\_struct

[T\\_ASW\\_cnf\\_struct](#) ASW\_cnf\_struct

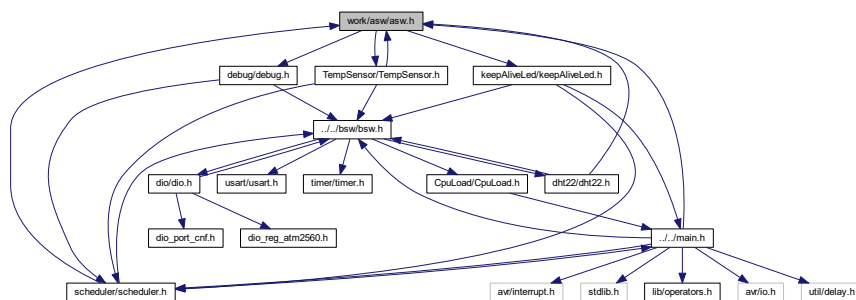
ASW configuration structure

Definition at line 17 of file asw.cpp.

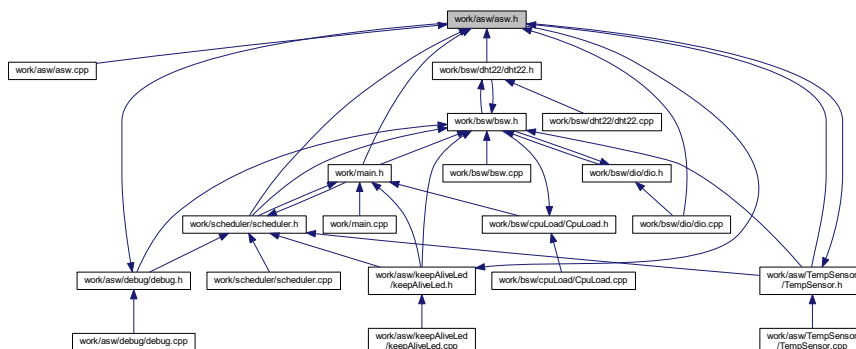
## 4.2 work/asw/asw.h File Reference

ASW main header file.

```
#include "debug/debug.h"
#include "keepAliveLed/keepAliveLed.h"
#include "TempSensor/TempSensor.h"
Include dependency graph for asw.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- struct [T\\_ASW\\_cnf\\_struct](#)  
ASW configuration structure.

## Functions

- void `asw_init()`  
*Initialization of ASW.*

## Variables

- `T_ASW_cnf_struct ASW_cnf_struct`

### 4.2.1 Detailed Description

ASW main header file.

#### Date

15 mars 2018

#### Author

nicls67

### 4.2.2 Function Documentation

#### 4.2.2.1 `asw_init()`

```
void asw_init ( )
```

Initialization of ASW.

This function instantiates all applicative objects. The addresses of objects are then stored in `ASW_cnf_struct` structure. This function shall be called after BSW initialization function.

#### Returns

Nothing

Definition at line 20 of file `asw.cpp`.

Here is the caller graph for this function:



### 4.2.3 Variable Documentation

#### 4.2.3.1 ASW\_cnf\_struct

`T_ASW_cnf_struct` ASW\_cnf\_struct

ASW configuration structure

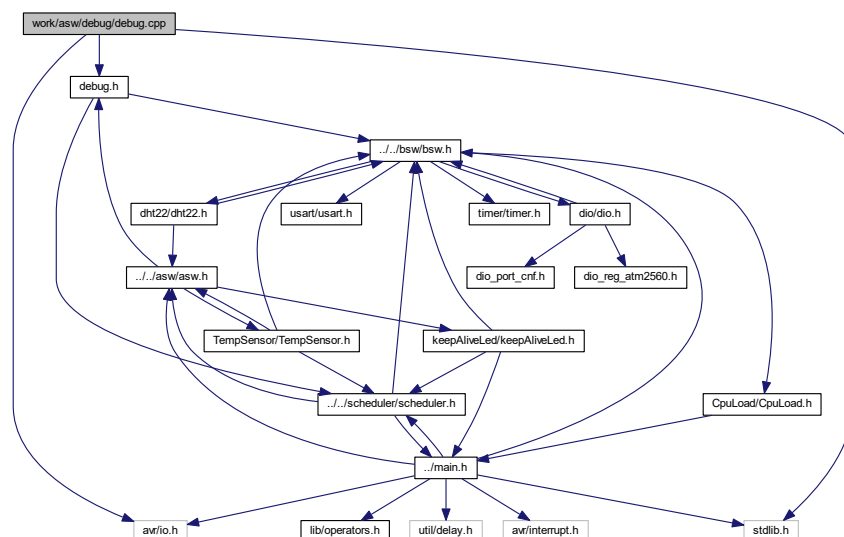
Definition at line 17 of file asw.cpp.

## 4.3 work/asw/debug/debug.cpp File Reference

This file defines classes for log and debug data transmission on USART link.

```
#include <avr/io.h>
#include <stdlib.h>
#include "debug.h"
```

Include dependency graph for debug.cpp:



### Variables

- const char `str_debug_main_menu` []  
Main menu of debug mode.

### 4.3.1 Detailed Description

This file defines classes for log and debug data transmission on USART link.

**Date**

15 mars 2018

**Author**

nicls67

### 4.3.2 Variable Documentation

#### 4.3.2.1 str\_debug\_main\_menu

```
const char str_debug_main_menu[ ]
```

**Initial value:**

```
=
    "\n\n"
    "Menu principal : \n"
    "1 : Afficher donnees capteurs\n"
    "2 : Afficher charge CPU\n"
    "\n"
    "s : Quitter debug\n"
```

Main menu of debug mode.

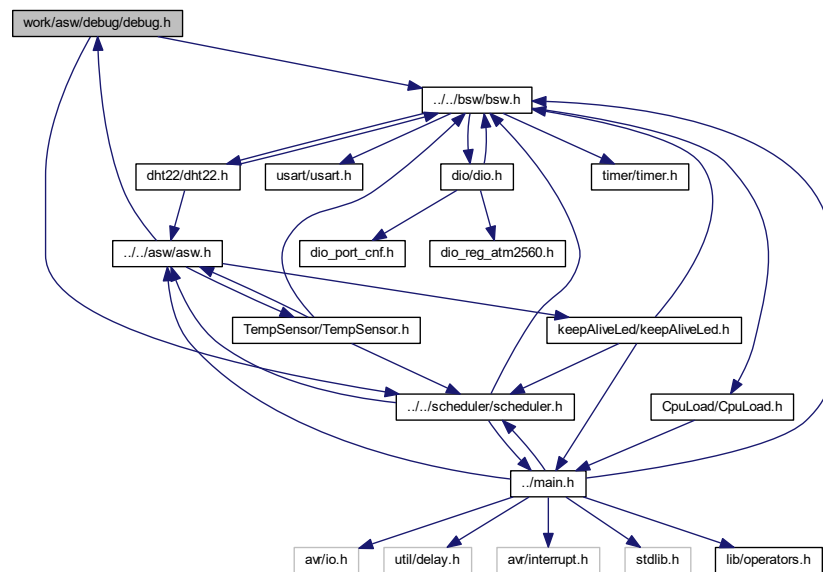
Definition at line 20 of file debug.cpp.

## 4.4 work/asw/debug/debug.h File Reference

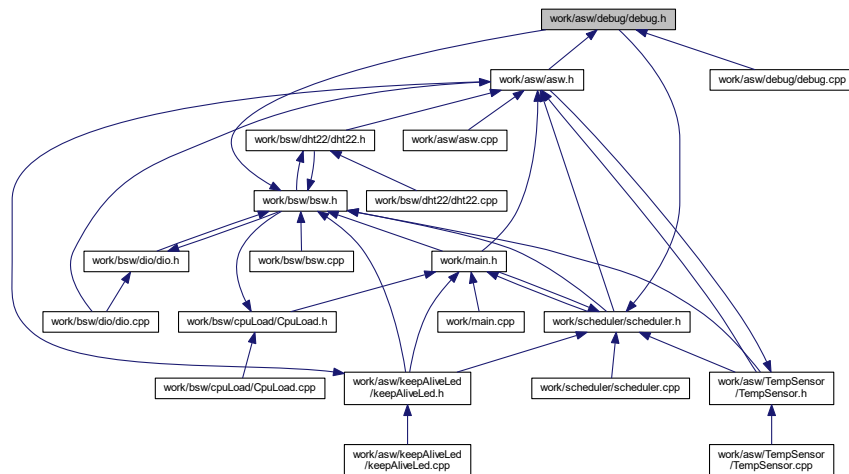
Header file for debug and logging functions.



```
#include "../bws/bws.h"
#include "../scheduler/scheduler.h"
Include dependency graph for debug.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class `UsartDebug`  
Class used for debugging on usart link.

## Macros

- `#define PERIOD_MS_TASK_DISPLAY_SENSORS 5000`
- `#define PERIOD_MS_TASK_DISPLAY_CPU_LOAD 5000`

## Enumerations

- enum `debug_state_t` { `INIT`, `WAIT_INIT`, `DISPLAY_DATA`, `DISPLAY_CPU_LOAD` }  
*Defines the debug states.*

### 4.4.1 Detailed Description

Header file for debug and logging functions.

#### Date

15 mars 2018

#### Author

nicls67

### 4.4.2 Macro Definition Documentation

#### 4.4.2.1 PERIOD\_MS\_TASK\_DISPLAY\_CPU\_LOAD

```
#define PERIOD_MS_TASK_DISPLAY_CPU_LOAD 5000
```

Period for displaying CPU load data

Definition at line 14 of file debug.h.

#### 4.4.2.2 PERIOD\_MS\_TASK\_DISPLAY\_SENSORS

```
#define PERIOD_MS_TASK_DISPLAY_SENSORS 5000
```

Period for displaying temperature and humidity data

Definition at line 13 of file debug.h.

### 4.4.3 Enumeration Type Documentation

#### 4.4.3.1 debug\_state\_t

```
enum debug_state_t
```

Defines the debug states.

## Enumerator

INIT	Init state : display the main menu
WAIT_INIT	Wait for a received character in init state
DISPLAY_DATA	Display sensor data in continuous
DISPLAY_CPU_LOAD	Display CPU load in continuous

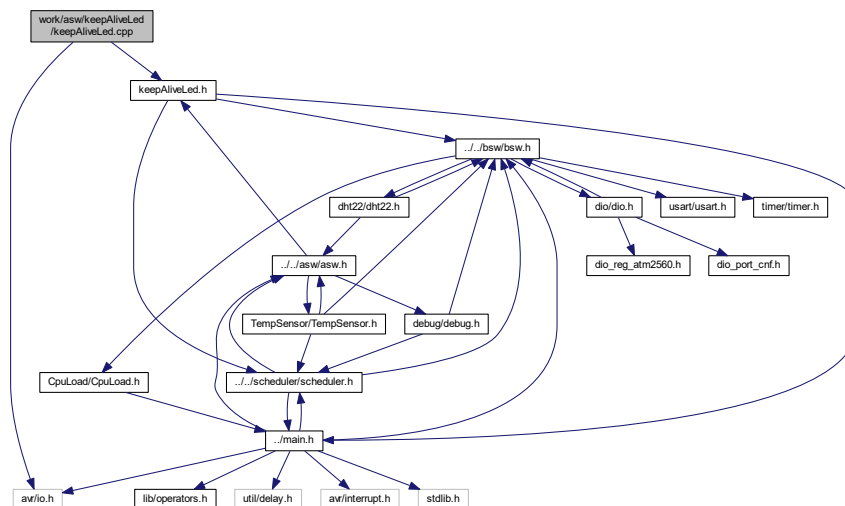
Definition at line 20 of file debug.h.

## 4.5 work/asw/keepAliveLed/keepAliveLed.cpp File Reference

Definition of function for class [keepAliveLed](#).

```
#include <avr/io.h>
#include "keepAliveLed.h"
```

Include dependency graph for keepAliveLed.cpp:



### 4.5.1 Detailed Description

Definition of function for class [keepAliveLed](#).

## Date

17 mars 2018

## Author

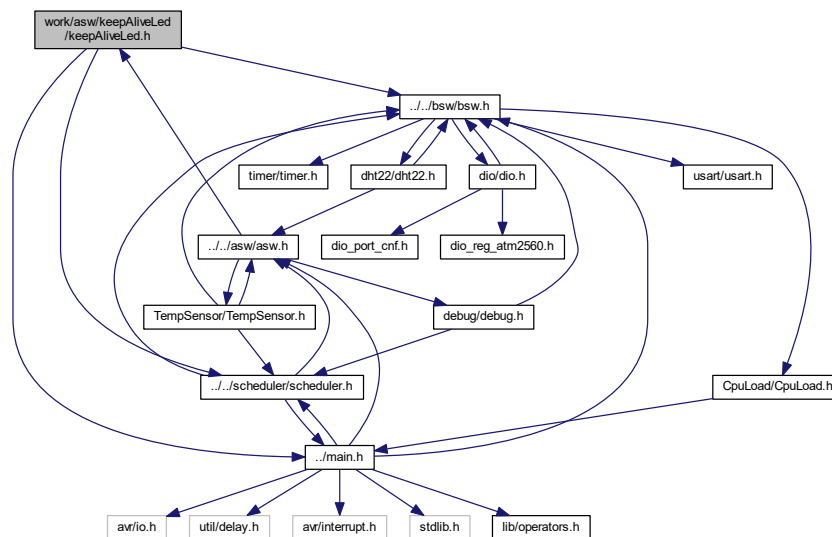
nicls67

## 4.6 work/asw/keepAliveLed/keepAliveLed.h File Reference

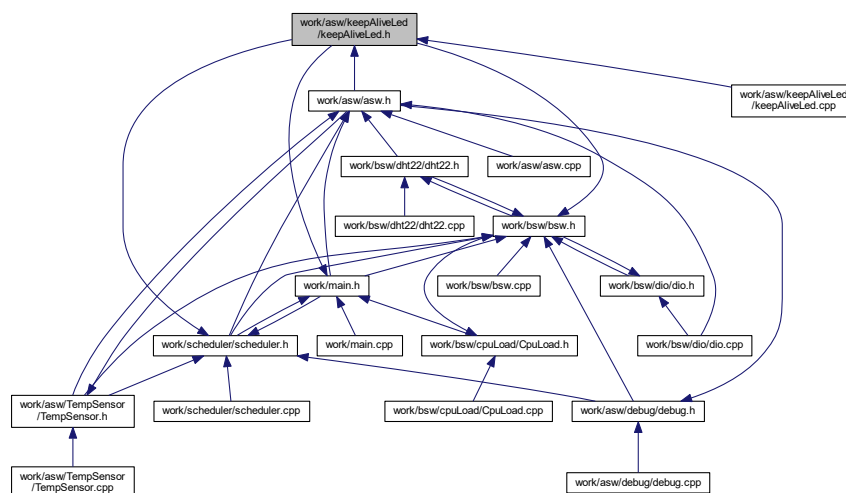
Class [keepAliveLed](#) header file.

```
#include "../bws/bws.h"
#include "../scheduler/scheduler.h"
#include "../main.h"
```

Include dependency graph for keepAliveLed.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [keepAliveLed](#)  
Class for keep-alive LED blinking.

## Macros

- `#define PERIOD_MS_TASK_LED SW_PERIOD_MS`
- `#define LED_PORT ENCODE_PORT(PORT_B, 7)`

### 4.6.1 Detailed Description

Class `keepAliveLed` header file.

#### Date

17 mars 2018

#### Author

nicls67

### 4.6.2 Macro Definition Documentation

#### 4.6.2.1 LED\_PORT

```
#define LED_PORT ENCODE_PORT(PORT_B, 7)
```

LED is connected to port PB7

Definition at line 16 of file `keepAliveLed.h`.

#### 4.6.2.2 PERIOD\_MS\_TASK\_LED

```
#define PERIOD_MS_TASK_LED SW_PERIOD_MS
```

Period for led blinking

Definition at line 15 of file `keepAliveLed.h`.

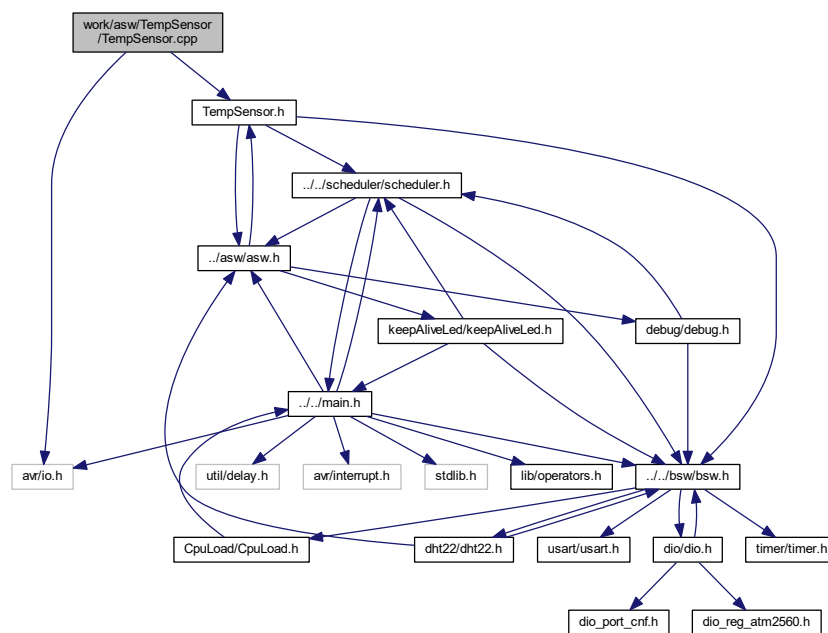
## 4.7 work/asw/TempSensor/TempSensor.cpp File Reference

Defines function of class [TempSensor](#).

```
#include <avr/io.h>
```

```
#include "TempSensor.h"
```

Include dependency graph for TempSensor.cpp:



### Macros

- `#define PIT_BEFORE_INVALID 60`

#### 4.7.1 Detailed Description

Defines function of class [TempSensor](#).

Date

23 mars 2018

Author

nicls67

#### 4.7.2 Macro Definition Documentation

## 4.7.2.1 PIT\_BEFORE\_INVALID

```
#define PIT_BEFORE_INVALID 60
```

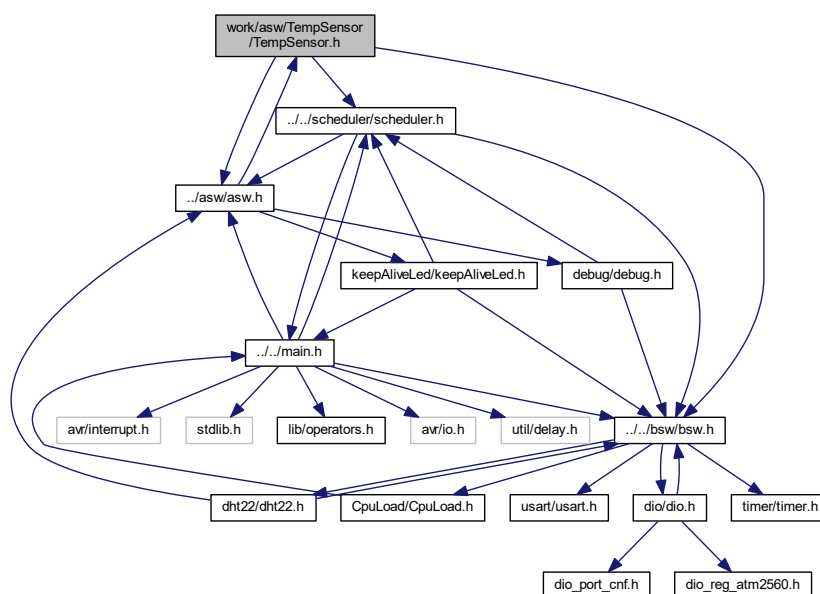
Definition at line 14 of file TempSensor.cpp.

## 4.8 work/asw/TempSensor/TempSensor.h File Reference

Class [TempSensor](#) header file.

```
#include "../scheduler/scheduler.h"
#include "../bsw/bsw.h"
#include "../asw.h"
```

Include dependency graph for TempSensor.h:







## 4.8.2.1 PERIOD\_MS\_TASK\_TEMP\_SENSOR

```
#define PERIOD_MS_TASK_TEMP_SENSOR 5000
```

Period for reading temperature data

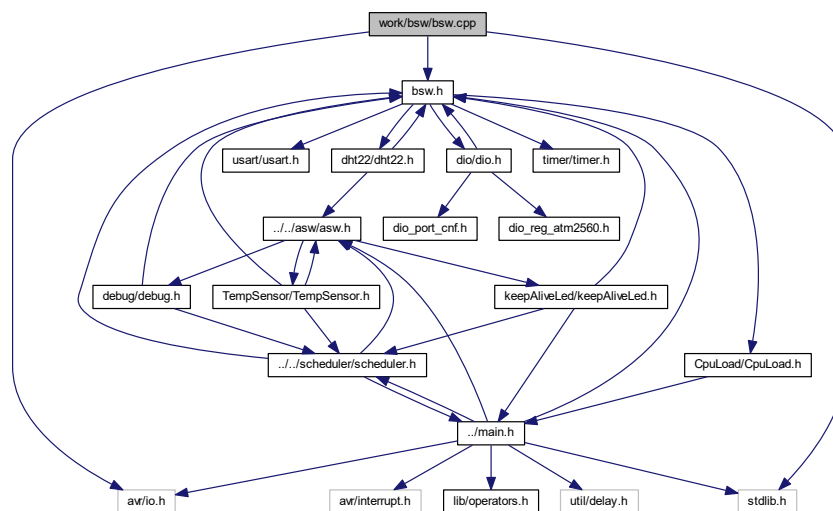
Definition at line 13 of file TempSensor.h.

## 4.9 work/bsw/bsw.cpp File Reference

BSW main file.

```
#include <avr/io.h>
#include <stdlib.h>
#include "bsw.h"
```

Include dependency graph for bsw.cpp:



## Functions

- void [bsw\\_init](#) ()  
*Initialization of BSW.*

## Variables

- [T\\_BSW\\_cnf\\_struct](#) [BSW\\_cnf\\_struct](#)

### 4.9.1 Detailed Description

BSW main file.

#### Date

13 mars 2018

#### Author

nicls67

### 4.9.2 Function Documentation

#### 4.9.2.1 bsw\_init()

```
void bsw_init ( )
```

Initialization of BSW.

This function instantiates all driver objects, leading hardware initialization. The addresses of driver objects are then stored in BSW\_cnf\_struct structure.

#### Returns

Nothing

Definition at line 18 of file bsw.cpp.

Here is the caller graph for this function:



### 4.9.3 Variable Documentation

## 4.9.3.1 BSW\_cnf\_struct

`T_BSW_cnf_struct` BSW\_cnf\_struct

BSW configuration structure

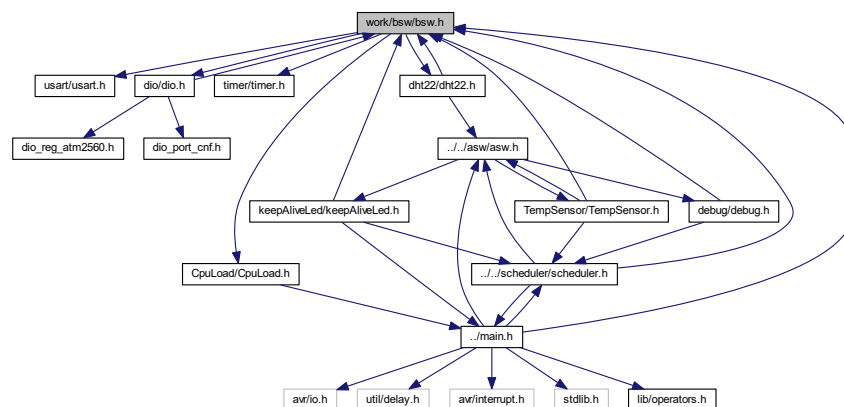
Definition at line 16 of file bsw.cpp.

## 4.10 work/bsw/bsw.h File Reference

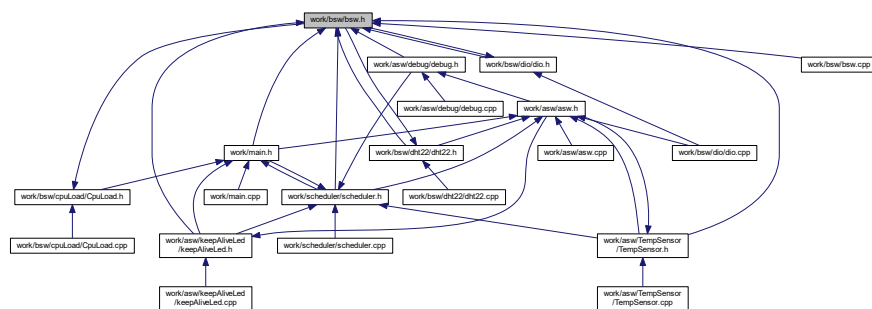
BSW main header file.

```
#include "usart/usart.h"
#include "dio/dio.h"
#include "timer/timer.h"
#include "dht22/dht22.h"
#include "CpuLoad/CpuLoad.h"
```

Include dependency graph for bsw.h:



This graph shows which files directly or indirectly include this file:



## Classes

- struct [T\\_BSW\\_cnf\\_struct](#)  
*BSW configuration structure.*

## Macros

- `#define USART\_BAUDRATE (uint16_t)9600`

## Functions

- void [bsw\\_init](#) ()  
*Initialization of BSW.*

## Variables

- [T\\_BSW\\_cnf\\_struct](#) [BSW\\_cnf\\_struct](#)

### 4.10.1 Detailed Description

BSW main header file.

#### Date

13 mars 2018

#### Author

nicls67

### 4.10.2 Macro Definition Documentation

#### 4.10.2.1 USART\_BAUDRATE

```
#define USART_BAUDRATE (uint16_t)9600
```

usart connection to PC uses a baud rate of 9600

Definition at line 24 of file bsw.h.

### 4.10.3 Function Documentation

#### 4.10.3.1 bsw\_init()

```
void bsw_init ( )
```

Initialization of BSW.

This function instantiates all driver objects, leading hardware initialization. The addresses of driver objects are then stored in BSW\_cnf\_struct structure.

##### Returns

Nothing

Definition at line 18 of file bsw.cpp.

Here is the caller graph for this function:



### 4.10.4 Variable Documentation

#### 4.10.4.1 BSW\_cnf\_struct

`T_BSW_cnf_struct` BSW\_cnf\_struct

BSW configuration structure

Definition at line 16 of file bsw.cpp.

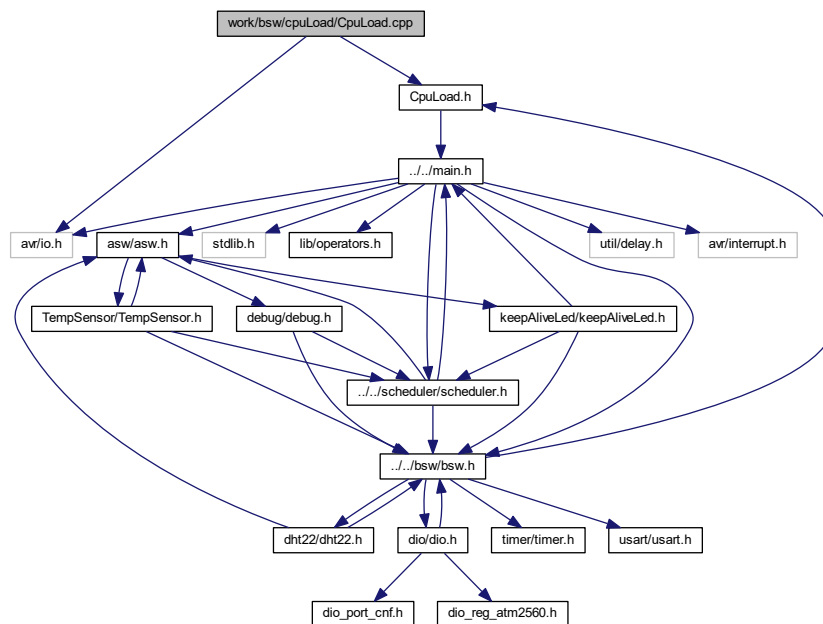
## 4.11 work/bsw/cpuLoad/CpuLoad.cpp File Reference

Defines functions of class [CpuLoad](#).

```
#include <avr/io.h>
```

```
#include "CpuLoad.h"
```

Include dependency graph for CpuLoad.cpp:



### 4.11.1 Detailed Description

Defines functions of class [CpuLoad](#).

Date

21 mars 2019

Author

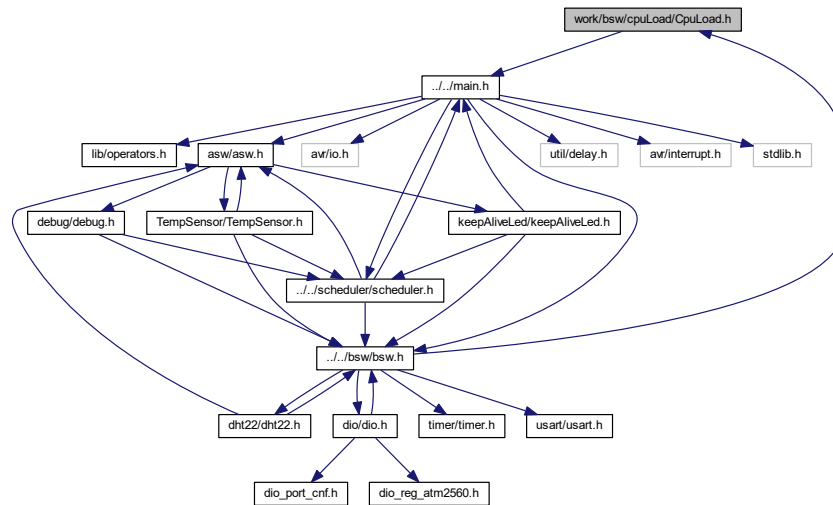
nicls67

## 4.12 work/bsw/cpuLoad/CpuLoad.h File Reference

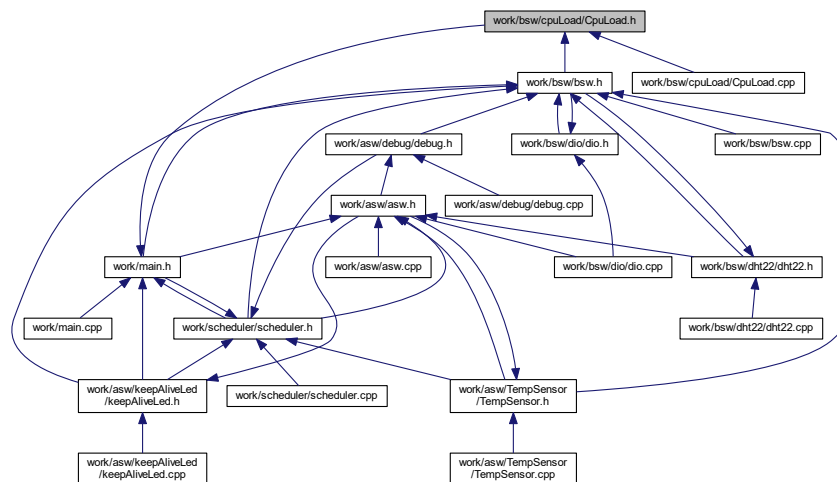
[CpuLoad](#) class header file.

```
#include "../..//main.h"
```

Include dependency graph for CpuLoad.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [CpuLoad](#)  
Class defining CPU load libraries.

## Macros

- `#define NB_OF_SAMPLES 50`

### 4.12.1 Detailed Description

[CpuLoad](#) class header file.

Date

21 mars 2019

Author

nicls67

### 4.12.2 Macro Definition Documentation

#### 4.12.2.1 NB\_OF\_SAMPLES

```
#define NB_OF_SAMPLES 50
```

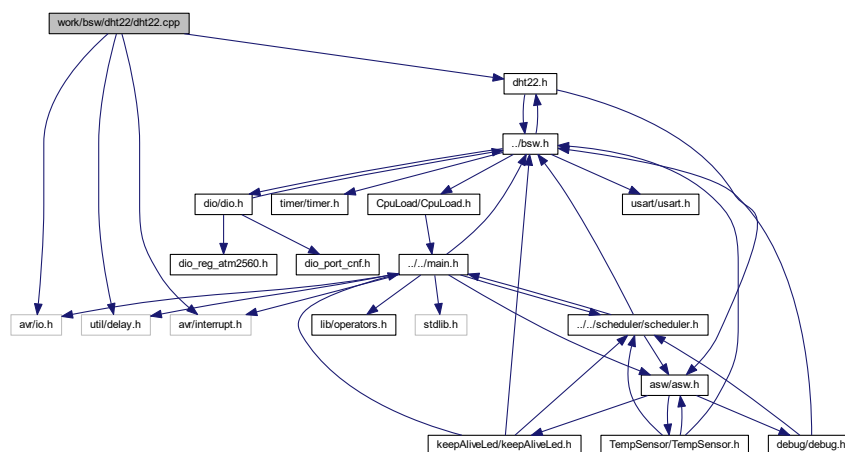
Definition at line 13 of file CpuLoad.h.

## 4.13 work/bsw/dht22/dht22.cpp File Reference

This file defines classes for DHT22 driver.

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "dht22.h"
```

Include dependency graph for dht22.cpp:





## Macros

- `#define MAX_WAIT_TIME_US 100`

### 4.13.1 Detailed Description

This file defines classes for DHT22 driver.

#### Date

23 mars 2018

#### Author

nicls67

### 4.13.2 Macro Definition Documentation

#### 4.13.2.1 MAX\_WAIT\_TIME\_US

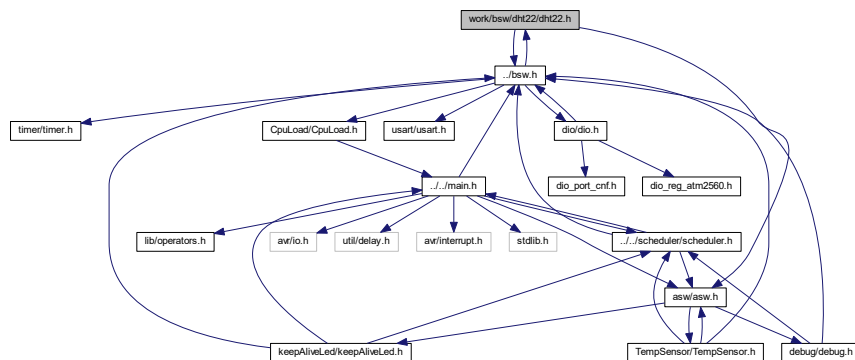
```
#define MAX_WAIT_TIME_US 100
```

Definition at line 20 of file dht22.cpp.

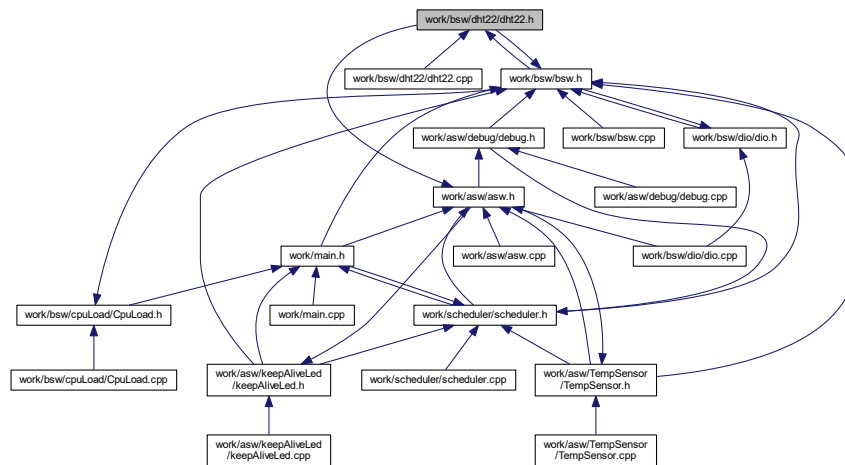
## 4.14 work/bsw/dht22/dht22.h File Reference

DHT22 driver header file.

```
#include "../bsw.h"
#include "../../asw/asw.h"
Include dependency graph for dht22.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [dht22](#)  
*DHT 22 driver class.*

## Macros

- `#define DHT22\_PORT ENCODE\_PORT(PORT\_B, 6)`

### 4.14.1 Detailed Description

DHT22 driver header file.

#### Date

23 mars 2018

#### Author

nicls67

### 4.14.2 Macro Definition Documentation

#### 4.14.2.1 DHT22\_PORT

```
#define DHT22_PORT ENCODE\_PORT(PORT\_B, 6)
```

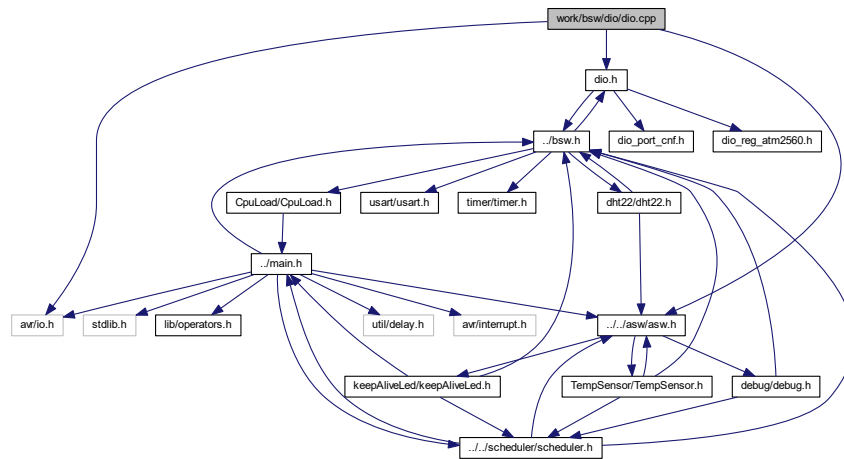
DHT22 is connected to port PB6

Definition at line 16 of file dht22.h.

## 4.15 work/bsw/dio/dio.cpp File Reference

DIO library.

```
#include <avr/io.h>
#include "dio.h"
#include "../asw/asw.h"
Include dependency graph for dio.cpp:
```



### 4.15.1 Detailed Description

DIO library.

Date

13 mars 2018

Author

nicls67

## 4.16 work/bsw/dio/dio.h File Reference

DIO library header file.

```
#include "../bsw.h"
#include "dio_port_conf.h"
```



### 4.16.1 Detailed Description

DIO library header file.

#### Date

13 mars 2018

#### Author

nicls67

### 4.16.2 Macro Definition Documentation

#### 4.16.2.1 DECODE\_PIN

```
#define DECODE_PIN(  
    portcode ) (uint8_t)(portcode & 0x7)
```

Macro used to extract pin index

Definition at line 19 of file dio.h.

#### 4.16.2.2 DECODE\_PORT

```
#define DECODE_PORT(  
    portcode ) (uint8_t)((portcode >> 3) & 0xF)
```

Macro used to extract port index

Definition at line 18 of file dio.h.

#### 4.16.2.3 ENCODE\_PORT

```
#define ENCODE_PORT(  
    port,  
    pin ) (uint8_t)((((uint8_t)(port & 0xF)) << 3) | (uint8_t)(pin & 0x7))
```

Macro used to encode port and pin indexes into one single byte

Definition at line 17 of file dio.h.

#### 4.16.2.4 PORT\_A

```
#define PORT_A 0
```

PORTA index

Definition at line 21 of file dio.h.

#### 4.16.2.5 PORT\_B

```
#define PORT_B 1
```

PORTB index

Definition at line 22 of file dio.h.

#### 4.16.2.6 PORT\_C

```
#define PORT_C 2
```

PORTC index

Definition at line 23 of file dio.h.

#### 4.16.2.7 PORT\_CNF\_IN

```
#define PORT_CNF_IN 0
```

Pin is configured as input

Definition at line 15 of file dio.h.

#### 4.16.2.8 PORT\_CNF\_OUT

```
#define PORT_CNF_OUT 1
```

Pin is configured as output

Definition at line 14 of file dio.h.

## 4.16.2.9 PORT\_D

```
#define PORT_D 3
```

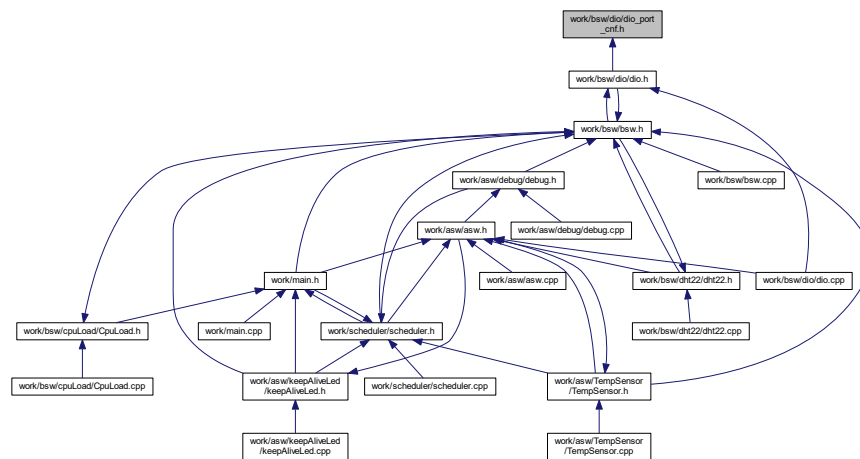
PORTD index

Definition at line 24 of file dio.h.

## 4.17 work/bsw/dio/dio\_port\_cnf.h File Reference

Digital ports configuration file.

This graph shows which files directly or indirectly include this file:



## Macros

- `#define PORTB_CNF_DDRB (uint8_t)0b11000000`  
Defines the configuration of DDRB register.
- `#define PORTB_CNF_PORTB (uint8_t)0b11000000`  
Defines the configuration of PORTB register.

## 4.17.1 Detailed Description

Digital ports configuration file.

Date

19 mars 2019

Author

nics67

## 4.17.2 Macro Definition Documentation

### 4.17.2.1 PORTB\_CNF\_DDRB

```
#define PORTB_CNF_DDRB (uint8_t)0b11000000
```

Defines the configuration of DDRB register.

This constant defines the direction of IO pins of PORT B. It will configure register DDRB.

PB0 : N/A  
PB1 : N/A  
PB2 : N/A  
PB3 : N/A  
PB4 : N/A  
PB5 : N/A  
PB6 : OUT  
PB7 : OUT

Definition at line 25 of file dio\_port\_cnf.h.

### 4.17.2.2 PORTB\_CNF\_PORTB

```
#define PORTB_CNF_PORTB (uint8_t)0b11000000
```

Defines the configuration of PORTB register.

This constant defines the initial value of IO pins for PORT B. It will configure register PORTB. Pins configured as input shall not be configured here.

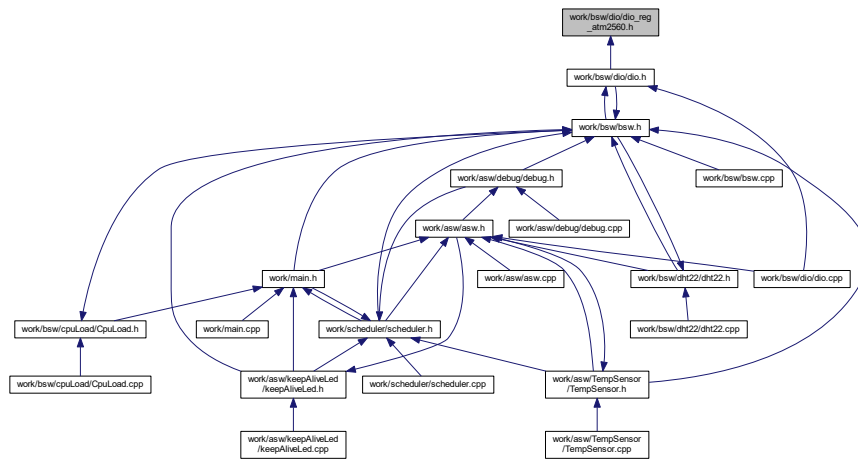
PB0 : N/A  
PB1 : N/A  
PB2 : N/A  
PB3 : N/A  
PB4 : N/A  
PB5 : N/A  
PB6 : HIGH  
PB7 : HIGH

Definition at line 40 of file dio\_port\_cnf.h.



## 4.18 work/bsw/dio/dio\_reg\_atm2560.h File Reference

This graph shows which files directly or indirectly include this file:



### Macros

- #define [PORTA\\_PTR](#) (volatile uint8\_t \*) (0x02 + 0x20)
- #define [PORTB\\_PTR](#) (volatile uint8\_t \*) (0x05 + 0x20)
- #define [PORTC\\_PTR](#) (volatile uint8\_t \*) (0x08 + 0x20)
- #define [PORTD\\_PTR](#) (volatile uint8\_t \*) (0x0B + 0x20)
- #define [PINA\\_PTR](#) (volatile uint8\_t \*) (0x00 + 0x20)
- #define [PINB\\_PTR](#) (volatile uint8\_t \*) (0x03 + 0x20)
- #define [PINC\\_PTR](#) (volatile uint8\_t \*) (0x06 + 0x20)
- #define [PIND\\_PTR](#) (volatile uint8\_t \*) (0x09 + 0x20)
- #define [DDRA\\_PTR](#) (volatile uint8\_t \*) (0x01 + 0x20)
- #define [DDRB\\_PTR](#) (volatile uint8\_t \*) (0x04 + 0x20)
- #define [DDRC\\_PTR](#) (volatile uint8\_t \*) (0x07 + 0x20)
- #define [DDRD\\_PTR](#) (volatile uint8\_t \*) (0x0A + 0x20)

### 4.18.1 Macro Definition Documentation

#### 4.18.1.1 DDRA\_PTR

```
#define DDRA_PTR (volatile uint8_t *) (0x01 + 0x20)
```

Macro defining pointer to DDR A register

Definition at line 24 of file dio\_reg\_atm2560.h.

#### 4.18.1.2 DDRB\_PTR

```
#define DDRB_PTR (volatile uint8_t *) (0x04 + 0x20)
```

Macro defining pointer to DDR B register

Definition at line 25 of file dio\_reg\_atm2560.h.

#### 4.18.1.3 DDRC\_PTR

```
#define DDRC_PTR (volatile uint8_t *) (0x07 + 0x20)
```

Macro defining pointer to DDR C register

Definition at line 26 of file dio\_reg\_atm2560.h.

#### 4.18.1.4 DDRD\_PTR

```
#define DDRD_PTR (volatile uint8_t *) (0x0A + 0x20)
```

Macro defining pointer to DDR D register

Definition at line 27 of file dio\_reg\_atm2560.h.

#### 4.18.1.5 PINA\_PTR

```
#define PINA_PTR (volatile uint8_t *) (0x00 + 0x20)
```

Macro defining pointer to PIN A register

Definition at line 19 of file dio\_reg\_atm2560.h.

#### 4.18.1.6 PINB\_PTR

```
#define PINB_PTR (volatile uint8_t *) (0x03 + 0x20)
```

Macro defining pointer to PIN B register

Definition at line 20 of file dio\_reg\_atm2560.h.

#### 4.18.1.7 PINC\_PTR

```
#define PINC_PTR (volatile uint8_t *) (0x06 + 0x20)
```

Macro defining pointer to PIN C register

Definition at line 21 of file dio\_reg\_atm2560.h.

#### 4.18.1.8 PIND\_PTR

```
#define PIND_PTR (volatile uint8_t *) (0x09 + 0x20)
```

Macro defining pointer to PIN D register

Definition at line 22 of file dio\_reg\_atm2560.h.

#### 4.18.1.9 PORTA\_PTR

```
#define PORTA_PTR (volatile uint8_t *) (0x02 + 0x20)
```

Macro defining pointer to PORT A register

Definition at line 14 of file dio\_reg\_atm2560.h.

#### 4.18.1.10 PORTB\_PTR

```
#define PORTB_PTR (volatile uint8_t *) (0x05 + 0x20)
```

Macro defining pointer to PORT B register

Definition at line 15 of file dio\_reg\_atm2560.h.

#### 4.18.1.11 PORTC\_PTR

```
#define PORTC_PTR (volatile uint8_t *) (0x08 + 0x20)
```

Macro defining pointer to PORT C register

Definition at line 16 of file dio\_reg\_atm2560.h.

#### 4.18.1.12 PORTD\_PTR

```
#define PORTD_PTR (volatile uint8_t *) (0x0B + 0x20)
```

Macro defining pointer to PORT D register

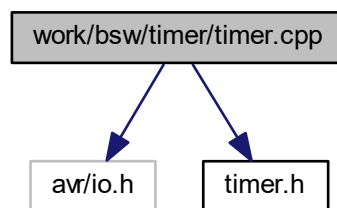
Definition at line 17 of file dio\_reg\_atm2560.h.

### 4.19 work/bsw/timer/timer.cpp File Reference

Defines function for class timer.

```
#include <avr/io.h>  
#include "timer.h"
```

Include dependency graph for timer.cpp:



#### 4.19.1 Detailed Description

Defines function for class timer.

Date

15 mars 2018

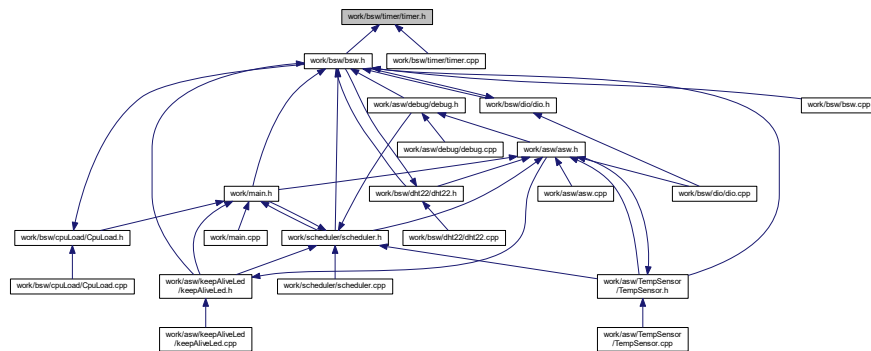
Author

nicls67

## 4.20 work/bsw/timer/timer.h File Reference

Timer class header file.

This graph shows which files directly or indirectly include this file:



### Classes

- class [timer](#)  
Class defining a timer.

#### 4.20.1 Detailed Description

Timer class header file.

#### Date

15 mars 2018

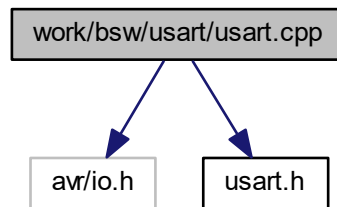
#### Author

nicls67

## 4.21 work/bsw/usart/usart.cpp File Reference

BSW library for USART.

```
#include <avr/io.h>
#include "usart.h"
Include dependency graph for usart.cpp:
```



#### 4.21.1 Detailed Description

BSW library for USART.

Date

13 mars 2018

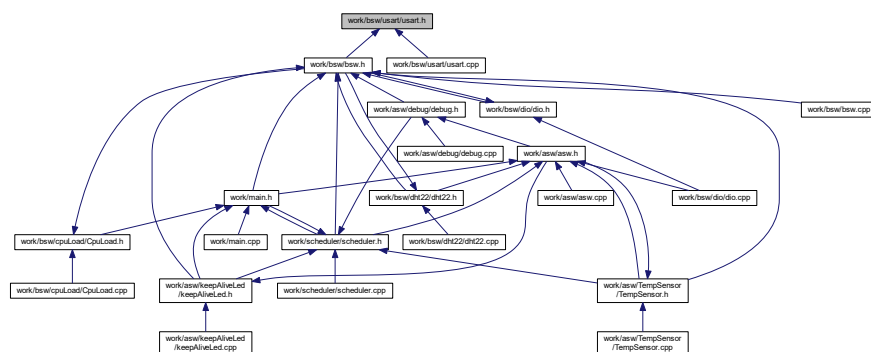
Author

nicls67

## 4.22 work/bsw/usart/usart.h File Reference

Header file for USART library.

This graph shows which files directly or indirectly include this file:



## Classes

- class `usart`  
*USART serial bus class.*

### 4.22.1 Detailed Description

Header file for USART library.

#### Date

13 mars 2018

#### Author

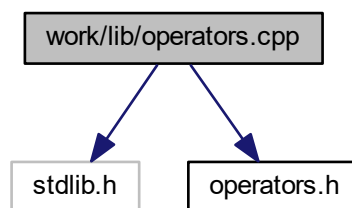
nicls67

## 4.23 work/lib/operators.cpp File Reference

c++ operators definitions

```
#include <stdlib.h>
#include "operators.h"
```

Include dependency graph for operators.cpp:



## Functions

- void \* `operator new` (size\_t a\_size)  
*Operator new.*
- void `operator delete` (void \*ptr)  
*Operator delete.*

### 4.23.1 Detailed Description

c++ operators definitions

Date

14 mars 2018

Author

nicls67

### 4.23.2 Function Documentation

#### 4.23.2.1 operator delete()

```
void operator delete (
    void * ptr )
```

Operator delete.

Equivalent to free function in C Free the memory zone at address ptr

Parameters

in	<i>ptr</i>	Pointer to the start of memory zone to free
----	------------	---

Returns

Nothing

Definition at line 18 of file operators.cpp.

#### 4.23.2.2 operator new()

```
void* operator new (
    size_t a_size )
```

Operator new.

Equivalent to malloc function in C Allocates a memory zone of size a\_size

Parameters

in	<i>a_size</i>	memory size to allocate
----	---------------	-------------------------



**Returns**

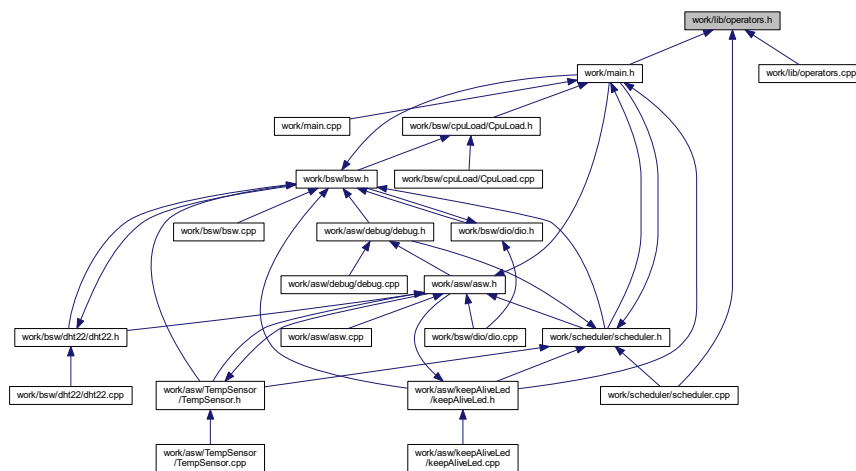
Pointer to the start of allocated memory zone

Definition at line 13 of file operators.cpp.

## 4.24 work/lib/operators.h File Reference

c++ operators definitions header file

This graph shows which files directly or indirectly include this file:

**Functions**

- void \* [operator new](#) (size\_t a\_size)  
*Operator new.*
- void [operator delete](#) (void \*ptr)  
*Operator delete.*

### 4.24.1 Detailed Description

c++ operators definitions header file

**Date**

14 mars 2018

**Author**

nicls67

## 4.24.2 Function Documentation

### 4.24.2.1 operator delete()

```
void operator delete (
    void * ptr )
```

Operator delete.

Equivalent to free function in C Free the memory zone at address ptr

#### Parameters

in	<i>ptr</i>	Pointer to the start of memory zone to free
----	------------	---

#### Returns

Nothing

Definition at line 18 of file operators.cpp.

### 4.24.2.2 operator new()

```
void* operator new (
    size_t a_size )
```

Operator new.

Equivalent to malloc function in C Allocates a memory zone of size a\_size

#### Parameters

in	<i>a_size</i>	memory size to allocate
----	---------------	-------------------------

#### Returns

Pointer to the start of allocated memory zone

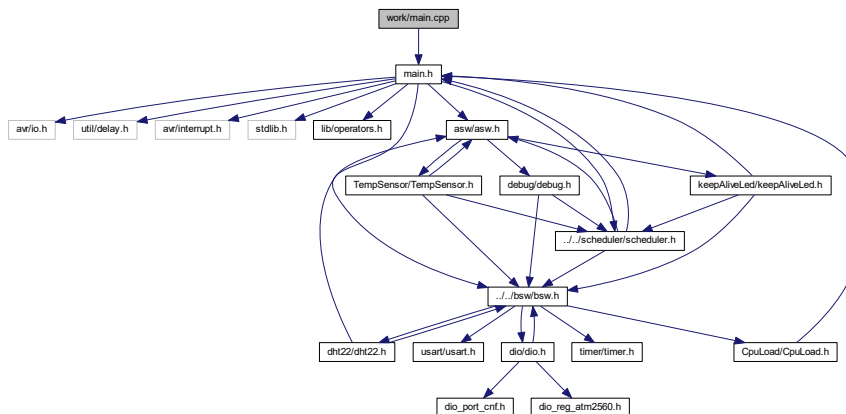
Definition at line 13 of file operators.cpp.

## 4.25 work/main.cpp File Reference

Background task file.

```
#include "main.h"
```

Include dependency graph for main.cpp:



## Functions

- [ISR](#) (TIMER1\_COMPA\_vect)  
*Main software interrupt.*
- [ISR](#) (USART0\_RX\_vect)  
*USART Rx Complete interrupt.*
- int [main](#) (void)  
*Background task of program.*

### 4.25.1 Detailed Description

Background task file.

Date

12 mars 2018

Author

nicls67

### 4.25.2 Function Documentation

#### 4.25.2.1 ISR() [1/2]

```
ISR (
    TIMER1_COMPA_vect )
```

Main software interrupt.

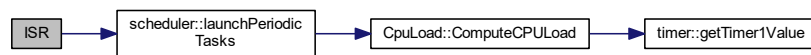
This function handles the interrupt raised by Timer #1. It wakes up the software every 500 ms to perform applications.

##### Returns

Nothing

Definition at line 19 of file main.cpp.

Here is the call graph for this function:



#### 4.25.2.2 ISR() [2/2]

```
ISR (
    USART0_RX_vect )
```

USART Rx Complete interrupt.

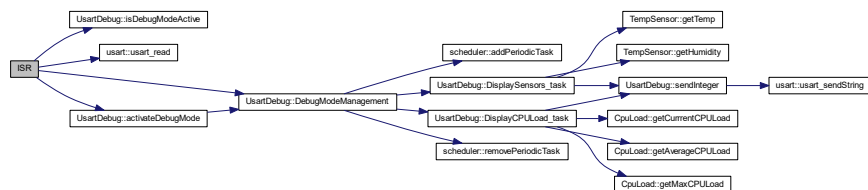
This function handles the interrupt raised when a frame has been received by USART. If debug mode mode is active, it calls debug mode management function. If inactive, it calls debug mode activation function if the received character is 'a'

##### Returns

Nothing

Definition at line 31 of file main.cpp.

Here is the call graph for this function:



## 4.25.2.3 main()

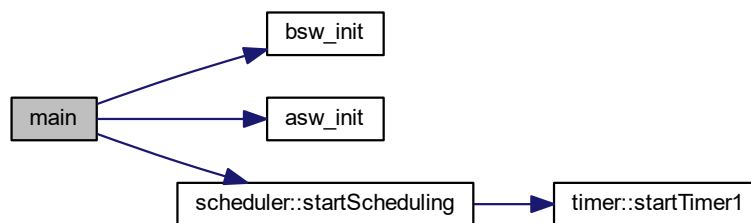
```
int main (
    void )
```

Background task of program.

This function initializes all the software and then goes into an infinite loop. Periodic interrupt will wake up the software to perform application

Definition at line 51 of file main.cpp.

Here is the call graph for this function:

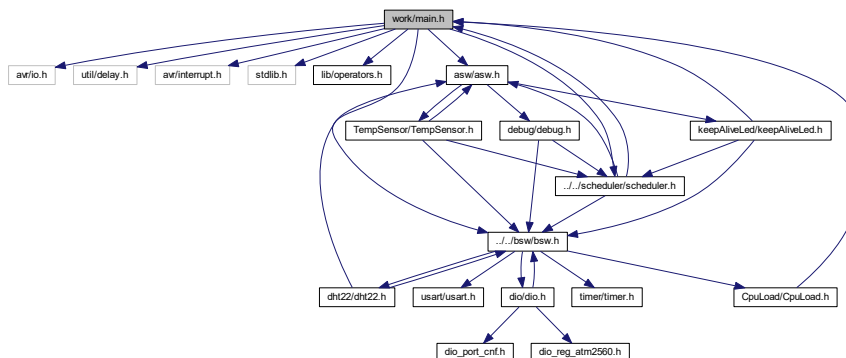


## 4.26 work/main.h File Reference

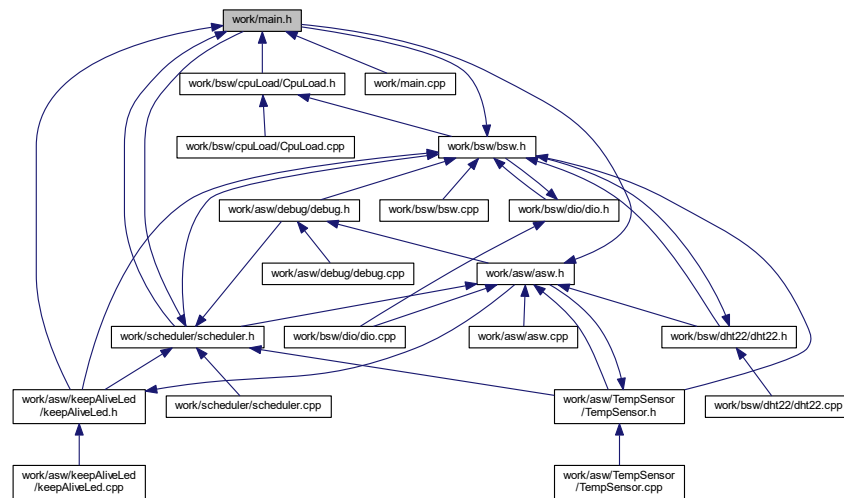
Background task header file.

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include "lib/operators.h"
#include "asw/asw.h"
#include "bsw/bsw.h"
#include "scheduler/scheduler.h"
```

Include dependency graph for main.h:



This graph shows which files directly or indirectly include this file:



#### 4.26.1 Detailed Description

Background task header file.

Date

17 mars 2018

Author

nicls67

#### 4.27 work/scheduler/scheduler.cpp File Reference

Defines scheduler class.

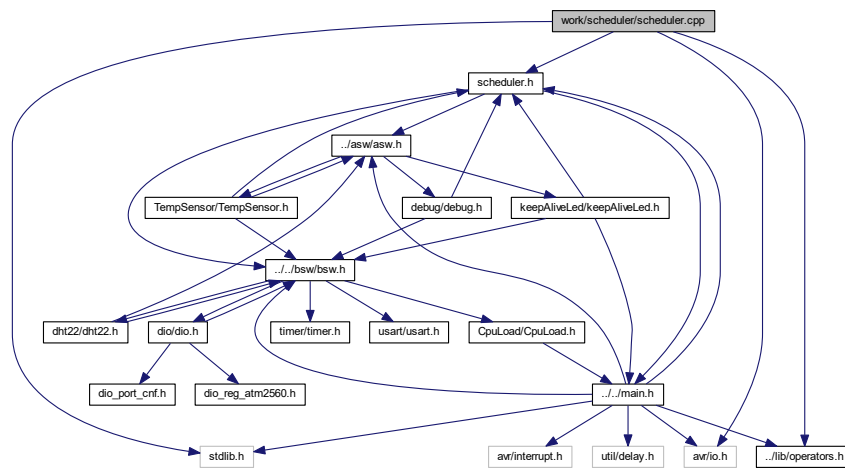
```

#include <stdlib.h>
#include <avr/io.h>
#include "../lib/operators.h"

```

```
#include "scheduler.h"
```

Include dependency graph for scheduler.cpp:



## Variables

- `scheduler * p_scheduler`

### 4.27.1 Detailed Description

Defines scheduler class.

#### Date

16 mars 2018

#### Author

nicls67

### 4.27.2 Variable Documentation

#### 4.27.2.1 p\_scheduler

```
scheduler* p_scheduler
```

Pointer to scheduler object

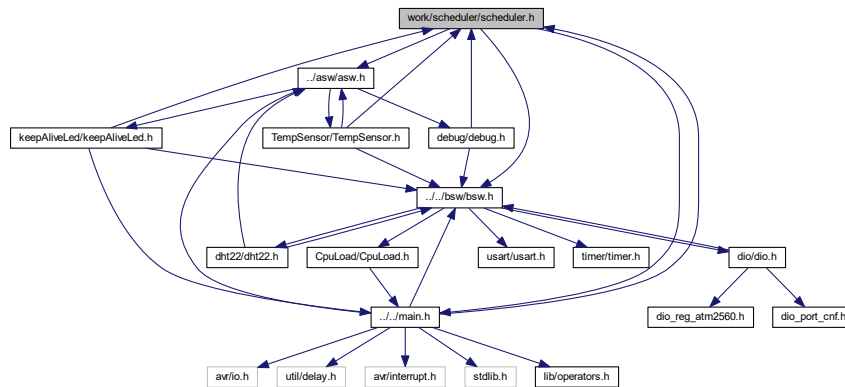
Definition at line 17 of file scheduler.cpp.

## 4.28 work/scheduler/scheduler.h File Reference

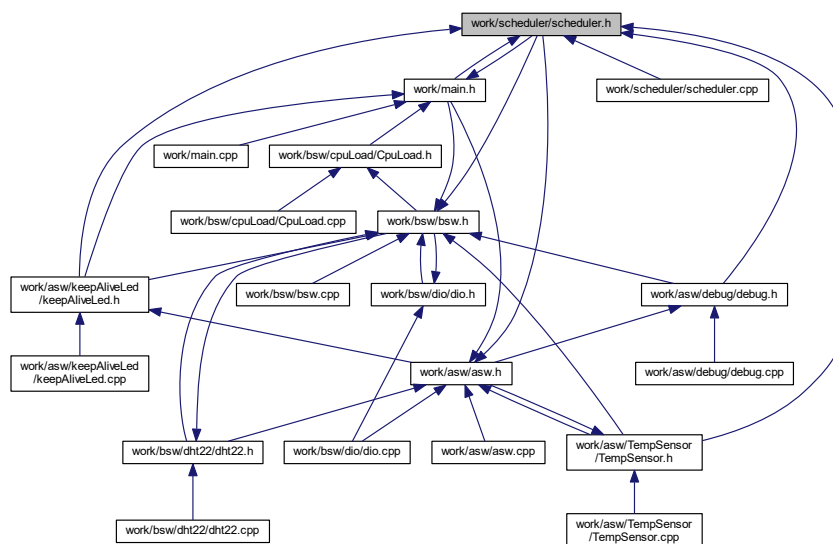
Scheduler class header file.

```
#include "../asw/asw.h"
#include "../bsw/bsw.h"
#include "../main.h"
```

Include dependency graph for scheduler.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [scheduler](#)  
*Scheduler class.*



## Macros

- `#define SW_PERIOD_MS 500`
- `#define PRESCALER_PERIODIC_TIMER 256`
- `#define TIMER CTC_VALUE ((F_CPU/PRESCALER_PERIODIC_TIMER)/(1000/SW_PERIOD_MS))`

## Typedefs

- `typedef void(* TaskPtr_t) (void)`  
*Type defining a pointer to function.*

## Variables

- `scheduler * p_scheduler`

### 4.28.1 Detailed Description

Scheduler class header file.

#### Date

16 mars 2018

#### Author

nicls67

### 4.28.2 Macro Definition Documentation

#### 4.28.2.1 PRESCALER\_PERIODIC\_TIMER

```
#define PRESCALER_PERIODIC_TIMER 256
```

Value of prescaler to use for periodic timer

Definition at line 19 of file scheduler.h.

#### 4.28.2.2 SW\_PERIOD\_MS

```
#define SW_PERIOD_MS 500
```

Software period, used to define periodic timer interrupt

Definition at line 18 of file scheduler.h.

#### 4.28.2.3 TIMER\_CTC\_VALUE

```
#define TIMER_CTC_VALUE ((F_CPU/PRESCALER_PERIODIC_TIMER)/(1000/SW_PERIOD_MS))
```

Compare value for periodic timer

Definition at line 20 of file scheduler.h.

### 4.28.3 Typedef Documentation

#### 4.28.3.1 TaskPtr\_t

```
typedef void(* TaskPtr_t) (void)
```

Type defining a pointer to function.

Definition at line 25 of file scheduler.h.

### 4.28.4 Variable Documentation

#### 4.28.4.1 p\_scheduler

```
scheduler* p_scheduler
```

Pointer to scheduler object

Definition at line 17 of file scheduler.cpp.

# Index

- ASW\_cnf\_struct
  - asw.cpp, [48](#)
  - asw.h, [51](#)
- activateDebugMode
  - UsartDebug, [39](#)
- addPeriodicTask
  - scheduler, [18](#)
- asw.cpp
  - ASW\_cnf\_struct, [48](#)
  - asw\_init, [48](#)
- asw.h
  - ASW\_cnf\_struct, [51](#)
  - asw\_init, [50](#)
- asw\_init
  - asw.cpp, [48](#)
  - asw.h, [50](#)
- BSW\_cnf\_struct
  - bsw.cpp, [62](#)
  - bsw.h, [65](#)
- blinkLed\_task
  - keepAliveLed, [16](#)
- bsw.cpp
  - BSW\_cnf\_struct, [62](#)
  - bsw\_init, [62](#)
- bsw.h
  - BSW\_cnf\_struct, [65](#)
  - bsw\_init, [64](#)
  - USART\_BAUDRATE, [64](#)
- bsw\_init
  - bsw.cpp, [62](#)
  - bsw.h, [64](#)
- ComputeCPULoad
  - CpuLoad, [6](#)
- configureTimer1
  - timer, [32](#)
- CpuLoad, [5](#)
  - ComputeCPULoad, [6](#)
  - CpuLoad, [5](#)
  - getAverageCPULoad, [6](#)
  - getCurrentCPULoad, [7](#)
  - getMaxCPULoad, [7](#)
- CpuLoad.h
  - NB\_OF\_SAMPLES, [68](#)
- DDRA\_PTR
  - dio\_reg\_atm2560.h, [77](#)
- DDRB\_PTR
  - dio\_reg\_atm2560.h, [77](#)
- DDRC\_PTR
  - dio\_reg\_atm2560.h, [78](#)
- DDRD\_PTR
  - dio\_reg\_atm2560.h, [78](#)
- DECODE\_PIN
  - dio.h, [73](#)
- DECODE\_PORT
  - dio.h, [73](#)
- DHT22\_PORT
  - dht22.h, [70](#)
- debug.cpp
  - str\_debug\_main\_menu, [52](#)
- debug.h
  - debug\_state\_t, [54](#)
  - PERIOD\_MS\_TASK\_DISPLAY\_CPU\_LOAD, [54](#)
  - PERIOD\_MS\_TASK\_DISPLAY\_SENSORS, [54](#)
- debug\_state\_t
  - debug.h, [54](#)
- DebugModeManagement
  - UsartDebug, [40](#)
- dht22, [8](#)
  - dht22, [9](#)
  - read, [9](#)
- dht22.cpp
  - MAX\_WAIT\_TIME\_US, [69](#)
- dht22.h
  - DHT22\_PORT, [70](#)
- dio, [10](#)
  - dio, [11](#)
  - dio\_changePortPinCnf, [11](#)
  - dio\_getPort, [12](#)
  - dio\_getPort\_fast, [12](#)
  - dio\_invertPort, [13](#)
  - dio\_memorizePINaddress, [13](#)
  - dio\_setPort, [14](#)
- dio.h
  - DECODE\_PIN, [73](#)
  - DECODE\_PORT, [73](#)
  - ENCODE\_PORT, [73](#)
  - PORT\_CNF\_IN, [74](#)
  - PORT\_CNF\_OUT, [74](#)
  - PORT\_A, [73](#)
  - PORT\_B, [74](#)
  - PORT\_C, [74](#)
  - PORT\_D, [74](#)
- dio\_changePortPinCnf
  - dio, [11](#)
- dio\_getPort
  - dio, [12](#)

- dio\_getPort\_fast
  - dio, [12](#)
- dio\_invertPort
  - dio, [13](#)
- dio\_memorizePINAddress
  - dio, [13](#)
- dio\_port\_cnf.h
  - PORTB\_CNF\_DDRB, [76](#)
  - PORTB\_CNF\_PORTB, [76](#)
- dio\_reg\_atm2560.h
  - DDRA\_PTR, [77](#)
  - DDRB\_PTR, [77](#)
  - DDRC\_PTR, [78](#)
  - DDRD\_PTR, [78](#)
  - PINA\_PTR, [78](#)
  - PINB\_PTR, [78](#)
  - PINC\_PTR, [78](#)
  - PIND\_PTR, [79](#)
  - PORTA\_PTR, [79](#)
  - PORTB\_PTR, [79](#)
  - PORTC\_PTR, [79](#)
  - PORTD\_PTR, [79](#)
- dio\_setPort
  - dio, [14](#)
- DisplayCPULoad\_task
  - UsartDebug, [41](#)
- DisplaySensors\_task
  - UsartDebug, [42](#)
- ENCODE\_PORT
  - dio.h, [73](#)
- getAverageCPULoad
  - CpuLoad, [6](#)
- getCurrrentCPULoad
  - CpuLoad, [7](#)
- getHumPtr
  - TempSensor, [27](#)
- getHumidity
  - TempSensor, [26](#)
- getMaxCPULoad
  - CpuLoad, [7](#)
- getPitNumber
  - scheduler, [19](#)
- getTemp
  - TempSensor, [27](#)
- getTempPtr
  - TempSensor, [28](#)
- getTimer1Value
  - timer, [33](#)
- ISR
  - main.cpp, [87](#), [88](#)
- isDebugModeActive
  - UsartDebug, [43](#)
- keepAliveLed, [15](#)
  - blinkLed\_task, [16](#)
  - keepAliveLed, [15](#)
- keepAliveLed.h
  - LED\_PORT, [57](#)
  - PERIOD\_MS\_TASK\_LED, [57](#)
- LED\_PORT
  - keepAliveLed.h, [57](#)
- launchPeriodicTasks
  - scheduler, [19](#)
- MAX\_WAIT\_TIME\_US
  - dht22.cpp, [69](#)
- main
  - main.cpp, [88](#)
- main.cpp
  - ISR, [87](#), [88](#)
  - main, [88](#)
- NB\_OF\_SAMPLES
  - CpuLoad.h, [68](#)
- operator delete
  - operators.cpp, [84](#)
  - operators.h, [86](#)
- operator new
  - operators.cpp, [84](#)
  - operators.h, [86](#)
- operators.cpp
  - operator delete, [84](#)
  - operator new, [84](#)
- operators.h
  - operator delete, [86](#)
  - operator new, [86](#)
- p\_TempSensor
  - T\_ASW\_cnf\_struct, [22](#)
- p\_cpuload
  - T\_BSW\_cnf\_struct, [24](#)
- p\_dht22
  - T\_BSW\_cnf\_struct, [24](#)
- p\_dio
  - T\_BSW\_cnf\_struct, [24](#)
- p\_keepAliveLed
  - T\_ASW\_cnf\_struct, [22](#)
- p\_scheduler
  - scheduler.cpp, [91](#)
  - scheduler.h, [94](#)
- p\_timer
  - T\_BSW\_cnf\_struct, [24](#)
- p\_usart
  - T\_BSW\_cnf\_struct, [24](#)
- p\_usartDebug
  - T\_ASW\_cnf\_struct, [23](#)
- PERIOD\_MS\_TASK\_DISPLAY\_CPU\_LOAD
  - debug.h, [54](#)
- PERIOD\_MS\_TASK\_DISPLAY\_SENSORS
  - debug.h, [54](#)
- PERIOD\_MS\_TASK\_LED
  - keepAliveLed.h, [57](#)
- PERIOD\_MS\_TASK\_TEMP\_SENSOR

- TempSensor.h, 60
- PINA\_PTR
  - dio\_reg\_atm2560.h, 78
- PINB\_PTR
  - dio\_reg\_atm2560.h, 78
- PINC\_PTR
  - dio\_reg\_atm2560.h, 78
- PIND\_PTR
  - dio\_reg\_atm2560.h, 79
- PIT\_BEFORE\_INVALID
  - TempSensor.cpp, 58
- PORT\_CNF\_IN
  - dio.h, 74
- PORT\_CNF\_OUT
  - dio.h, 74
- PORT\_A
  - dio.h, 73
- PORT\_B
  - dio.h, 74
- PORT\_C
  - dio.h, 74
- PORT\_D
  - dio.h, 74
- PORTA\_PTR
  - dio\_reg\_atm2560.h, 79
- PORTB\_CNF\_DDRB
  - dio\_port\_cnf.h, 76
- PORTB\_CNF\_PORTB
  - dio\_port\_cnf.h, 76
- PORTB\_PTR
  - dio\_reg\_atm2560.h, 79
- PORTC\_PTR
  - dio\_reg\_atm2560.h, 79
- PORTD\_PTR
  - dio\_reg\_atm2560.h, 79
- PRESCALER\_PERIODIC\_TIMER
  - scheduler.h, 93
- read
  - dht22, 9
- readTempSensor\_task
  - TempSensor, 28
- removePeriodicTask
  - scheduler, 20
- SW\_PERIOD\_MS
  - scheduler.h, 93
- scheduler, 17
  - addPeriodicTask, 18
  - getPitNumber, 19
  - launchPeriodicTasks, 19
  - removePeriodicTask, 20
  - scheduler, 18
  - startScheduling, 21
- scheduler.cpp
  - p\_scheduler, 91
- scheduler.h
  - p\_scheduler, 94
  - PRESCALER\_PERIODIC\_TIMER, 93
  - SW\_PERIOD\_MS, 93
  - TIMER\_CTC\_VALUE, 93
  - TaskPtr\_t, 94
- sendBool
  - UsartDebug, 43
- sendInteger
  - UsartDebug, 44
- setBaudRate
  - usart, 36
- setValidity
  - TempSensor, 29
- startScheduling
  - scheduler, 21
- startTimer1
  - timer, 33
- stopTimer1
  - timer, 34
- str\_debug\_main\_menu
  - debug.cpp, 52
- T\_ASW\_cnf\_struct, 22
  - p\_TempSensor, 22
  - p\_keepAliveLed, 22
  - p\_usartDebug, 23
- T\_BSW\_cnf\_struct, 23
  - p\_cpuload, 24
  - p\_dht22, 24
  - p\_dio, 24
  - p\_timer, 24
  - p\_usart, 24
- TIMER\_CTC\_VALUE
  - scheduler.h, 93
- TaskPtr\_t
  - scheduler.h, 94
- TempSensor, 25
  - getHumPtr, 27
  - getHumidity, 26
  - getTemp, 27
  - getTempPtr, 28
  - readTempSensor\_task, 28
  - setValidity, 29
  - TempSensor, 26
  - updateLastValidValues, 30
- TempSensor.cpp
  - PIT\_BEFORE\_INVALID, 58
- TempSensor.h
  - PERIOD\_MS\_TASK\_TEMP\_SENSOR, 60
- timer, 31
  - configureTimer1, 32
  - getTimer1Value, 33
  - startTimer1, 33
  - stopTimer1, 34
  - timer, 31
- USART\_BAUDRATE
  - bsw.h, 64
- updateLastValidValues
  - TempSensor, 30
- usart, 34

- setBaudRate, [36](#)
- usart, [35](#)
- usart\_init, [36](#)
- usart\_read, [37](#)
- usart\_sendString, [37](#)
- usart\_init
  - usart, [36](#)
- usart\_read
  - usart, [37](#)
- usart\_sendString
  - usart, [37](#)
- UsartDebug, [38](#)
  - activateDebugMode, [39](#)
  - DebugModeManagement, [40](#)
  - DisplayCPULoad\_task, [41](#)
  - DisplaySensors\_task, [42](#)
  - isDebugModeActive, [43](#)
  - sendBool, [43](#)
  - sendInteger, [44](#)
  - UsartDebug, [39](#)
- work/asw/TempSensor/TempSensor.cpp, [58](#)
- work/asw/TempSensor/TempSensor.h, [59](#)
- work/asw/asw.cpp, [47](#)
- work/asw/asw.h, [49](#)
- work/asw/debug/debug.cpp, [51](#)
- work/asw/debug/debug.h, [52](#)
- work/asw/keepAliveLed/keepAliveLed.cpp, [55](#)
- work/asw/keepAliveLed/keepAliveLed.h, [56](#)
- work/bsw/bsw.cpp, [61](#)
- work/bsw/bsw.h, [63](#)
- work/bsw/cpuLoad/CpuLoad.cpp, [66](#)
- work/bsw/cpuLoad/CpuLoad.h, [66](#)
- work/bsw/dht22/dht22.cpp, [68](#)
- work/bsw/dht22/dht22.h, [69](#)
- work/bsw/dio/dio.cpp, [71](#)
- work/bsw/dio/dio.h, [71](#)
- work/bsw/dio/dio\_port\_conf.h, [75](#)
- work/bsw/dio/dio\_reg\_atm2560.h, [77](#)
- work/bsw/timer/timer.cpp, [80](#)
- work/bsw/timer/timer.h, [81](#)
- work/bsw/usart/usart.cpp, [81](#)
- work/bsw/usart/usart.h, [82](#)
- work/lib/operators.cpp, [83](#)
- work/lib/operators.h, [85](#)
- work/main.cpp, [86](#)
- work/main.h, [89](#)
- work/scheduler/scheduler.cpp, [90](#)
- work/scheduler/scheduler.h, [92](#)