

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	dht22 Class Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Constructor & Destructor Documentation . . . . .	5
3.1.2.1	dht22() . . . . .	5
3.1.3	Member Function Documentation . . . . .	6
3.1.3.1	read() . . . . .	6
3.2	dio Class Reference . . . . .	7
3.2.1	Detailed Description . . . . .	7
3.2.2	Constructor & Destructor Documentation . . . . .	7
3.2.2.1	dio() . . . . .	8
3.2.3	Member Function Documentation . . . . .	8
3.2.3.1	dio_changePortPinCnf() . . . . .	8
3.2.3.2	dio_getPort() . . . . .	9
3.2.3.3	dio_getPort_fast() . . . . .	9
3.2.3.4	dio_invertPort() . . . . .	9
3.2.3.5	dio_memorizePINaddress() . . . . .	10
3.2.3.6	dio_setPort() . . . . .	11

3.3	keepAliveLed Class Reference	11
3.3.1	Detailed Description	12
3.3.2	Constructor & Destructor Documentation	12
3.3.2.1	keepAliveLed()	12
3.3.3	Member Function Documentation	12
3.3.3.1	blinkLed_task()	13
3.4	scheduler Class Reference	13
3.4.1	Detailed Description	14
3.4.2	Constructor & Destructor Documentation	14
3.4.2.1	scheduler()	14
3.4.3	Member Function Documentation	15
3.4.3.1	addPeriodicTask()	15
3.4.3.2	getPitNumber()	15
3.4.3.3	launchPeriodicTasks()	16
3.4.3.4	removePeriodicTask()	16
3.4.3.5	startScheduling()	17
3.5	T_ASW_cnf_struct Struct Reference	18
3.5.1	Detailed Description	18
3.5.2	Member Data Documentation	19
3.5.2.1	p_keepAliveLed	19
3.5.2.2	p_TempSensor	19
3.5.2.3	p_usartDebug	19
3.6	T_BSW_cnf_struct Struct Reference	19
3.6.1	Detailed Description	20
3.6.2	Member Data Documentation	20
3.6.2.1	p_dht22	20
3.6.2.2	p_dio	20
3.6.2.3	p_timer	20
3.6.2.4	p_usart	21
3.7	TempSensor Class Reference	21

3.7.1	Detailed Description	21
3.7.2	Constructor & Destructor Documentation	21
3.7.2.1	TempSensor()	22
3.7.3	Member Function Documentation	22
3.7.3.1	getHumidity()	22
3.7.3.2	getHumPtr()	23
3.7.3.3	getTemp()	23
3.7.3.4	getTempPtr()	24
3.7.3.5	readTempSensor_task()	24
3.7.3.6	setValidity()	25
3.7.3.7	updateLastValidValues()	26
3.8	timer Class Reference	26
3.8.1	Detailed Description	27
3.8.2	Constructor & Destructor Documentation	27
3.8.2.1	timer()	27
3.8.3	Member Function Documentation	27
3.8.3.1	configureTimer1()	27
3.8.3.2	startTimer1()	28
3.8.3.3	stopTimer1()	29
3.9	usart Class Reference	29
3.9.1	Detailed Description	29
3.9.2	Constructor & Destructor Documentation	29
3.9.2.1	usart()	29
3.9.3	Member Function Documentation	30
3.9.3.1	setBaudRate()	30
3.9.3.2	usart_init()	31
3.9.3.3	usart_read()	31
3.9.3.4	usart_sendString()	32
3.10	UsartDebug Class Reference	32
3.10.1	Detailed Description	33
3.10.2	Constructor & Destructor Documentation	33
3.10.2.1	UsartDebug()	33
3.10.3	Member Function Documentation	33
3.10.3.1	activateDebugMode()	34
3.10.3.2	DebugModeManagement()	34
3.10.3.3	DisplaySensors_task()	35
3.10.3.4	isDebugModeActive()	36
3.10.3.5	sendBool()	37
3.10.3.6	sendData()	37
3.10.3.7	sendInteger()	38

<b>4</b>	<b>File Documentation</b>	<b>41</b>
4.1	work/asw/asw.cpp File Reference	41
4.1.1	Detailed Description	42
4.1.2	Function Documentation	42
4.1.2.1	asw_init()	42
4.1.3	Variable Documentation	42
4.1.3.1	ASW_cnf_struct	43
4.2	work/asw/asw.h File Reference	43
4.2.1	Detailed Description	44
4.2.2	Function Documentation	44
4.2.2.1	asw_init()	44
4.2.3	Variable Documentation	45
4.2.3.1	ASW_cnf_struct	45
4.3	work/asw/debug/debug.cpp File Reference	45
4.3.1	Detailed Description	46
4.3.2	Variable Documentation	46
4.3.2.1	str_debug_main_menu	46
4.4	work/asw/debug/debug.h File Reference	46
4.4.1	Detailed Description	47
4.4.2	Macro Definition Documentation	48
4.4.2.1	PERIOD_MS_TASK_DISPLAY_SENSORS	48
4.4.3	Enumeration Type Documentation	48
4.4.3.1	debug_state_t	48
4.5	work/asw/keepAliveLed/keepAliveLed.cpp File Reference	48
4.5.1	Detailed Description	49
4.6	work/asw/keepAliveLed/keepAliveLed.h File Reference	49
4.6.1	Detailed Description	51
4.6.2	Macro Definition Documentation	51
4.6.2.1	LED_PORT	51
4.6.2.2	PERIOD_MS_TASK_LED	51

4.7	work/asw/TempSensor/TempSensor.cpp File Reference	52
4.7.1	Detailed Description	52
4.7.2	Macro Definition Documentation	52
4.7.2.1	PIT_BEFORE_INVALID	53
4.8	work/asw/TempSensor/TempSensor.h File Reference	53
4.8.1	Detailed Description	54
4.8.2	Macro Definition Documentation	54
4.8.2.1	PERIOD_MS_TASK_TEMP_SENSOR	55
4.9	work/bsw/bsw.cpp File Reference	55
4.9.1	Detailed Description	55
4.9.2	Function Documentation	56
4.9.2.1	bsw_init()	56
4.9.3	Variable Documentation	56
4.9.3.1	BSW_cnf_struct	56
4.10	work/bsw/bsw.h File Reference	57
4.10.1	Detailed Description	58
4.10.2	Macro Definition Documentation	58
4.10.2.1	USART_BAUDRATE	58
4.10.3	Function Documentation	58
4.10.3.1	bsw_init()	59
4.10.4	Variable Documentation	59
4.10.4.1	BSW_cnf_struct	59
4.11	work/bsw/dht22/dht22.cpp File Reference	60
4.11.1	Detailed Description	60
4.11.2	Macro Definition Documentation	60
4.11.2.1	MAX_WAIT_TIME_US	61
4.12	work/bsw/dht22/dht22.h File Reference	61
4.12.1	Detailed Description	62
4.12.2	Macro Definition Documentation	62
4.12.2.1	DHT22_PORT	62

4.13	work/bsw/dio/dio.cpp File Reference	63
4.13.1	Detailed Description	63
4.14	work/bsw/dio/dio.h File Reference	63
4.14.1	Detailed Description	65
4.14.2	Macro Definition Documentation	65
4.14.2.1	DECODE_PIN	65
4.14.2.2	DECODE_PORT	65
4.14.2.3	ENCODE_PORT	66
4.14.2.4	PORT_A	66
4.14.2.5	PORT_B	66
4.14.2.6	PORT_C	66
4.14.2.7	PORT_CNF_IN	66
4.14.2.8	PORT_CNF_OUT	67
4.14.2.9	PORT_D	67
4.15	work/bsw/dio/dio_port_cnf.h File Reference	67
4.15.1	Detailed Description	68
4.15.2	Macro Definition Documentation	68
4.15.2.1	PORTB_CNF_DDRB	68
4.15.2.2	PORTB_CNF_PORTB	68
4.16	work/bsw/dio/dio_reg_atm2560.h File Reference	69
4.16.1	Macro Definition Documentation	69
4.16.1.1	DDRA_PTR	69
4.16.1.2	DDRB_PTR	70
4.16.1.3	DDRC_PTR	70
4.16.1.4	DDRD_PTR	70
4.16.1.5	PINA_PTR	70
4.16.1.6	PINB_PTR	70
4.16.1.7	PINC_PTR	71
4.16.1.8	PIND_PTR	71
4.16.1.9	PORTA_PTR	71



4.16.1.10 PORTB_PTR . . . . .	71
4.16.1.11 PORTC_PTR . . . . .	71
4.16.1.12 PORTD_PTR . . . . .	72
4.17 work/bsw/timer/timer.cpp File Reference . . . . .	72
4.17.1 Detailed Description . . . . .	72
4.18 work/bsw/timer/timer.h File Reference . . . . .	73
4.18.1 Detailed Description . . . . .	73
4.19 work/bsw/usart/usart.cpp File Reference . . . . .	73
4.19.1 Detailed Description . . . . .	74
4.20 work/bsw/usart/usart.h File Reference . . . . .	74
4.20.1 Detailed Description . . . . .	75
4.21 work/lib/operators.cpp File Reference . . . . .	75
4.21.1 Detailed Description . . . . .	76
4.21.2 Function Documentation . . . . .	76
4.21.2.1 operator delete() . . . . .	76
4.21.2.2 operator new() . . . . .	76
4.22 work/lib/operators.h File Reference . . . . .	77
4.22.1 Detailed Description . . . . .	77
4.22.2 Function Documentation . . . . .	77
4.22.2.1 operator delete() . . . . .	78
4.22.2.2 operator new() . . . . .	78
4.23 work/main.cpp File Reference . . . . .	78
4.23.1 Detailed Description . . . . .	79
4.23.2 Function Documentation . . . . .	79
4.23.2.1 ISR() [1/2] . . . . .	80
4.23.2.2 ISR() [2/2] . . . . .	80
4.23.2.3 main() . . . . .	81
4.24 work/main.h File Reference . . . . .	81
4.24.1 Detailed Description . . . . .	82
4.25 work/scheduler/scheduler.cpp File Reference . . . . .	83
4.25.1 Detailed Description . . . . .	83
4.25.2 Variable Documentation . . . . .	83
4.25.2.1 p_scheduler . . . . .	84
4.26 work/scheduler/scheduler.h File Reference . . . . .	84
4.26.1 Detailed Description . . . . .	85
4.26.2 Macro Definition Documentation . . . . .	85
4.26.2.1 PRESCALER_PERIODIC_TIMER . . . . .	85
4.26.2.2 SW_PERIOD_MS . . . . .	86
4.26.2.3 TIMER_CTC_VALUE . . . . .	86
4.26.3 Typedef Documentation . . . . .	86
4.26.3.1 TaskPtr_t . . . . .	86
4.26.4 Variable Documentation . . . . .	86
4.26.4.1 p_scheduler . . . . .	86



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">dht22</a>	DHT 22 driver class . . . . .	5
<a href="#">dio</a>	DIO class . . . . .	7
<a href="#">keepAliveLed</a>	Class for keep-alive LED blinking . . . . .	11
<a href="#">scheduler</a>	Scheduler class . . . . .	13
<a href="#">T_ASW_cnf_struct</a>	ASW configuration structure . . . . .	18
<a href="#">T_BSW_cnf_struct</a>	BSW configuration structure . . . . .	19
<a href="#">TempSensor</a>	. . . . .	21
<a href="#">timer</a>	Class defining a timer . . . . .	26
<a href="#">usart</a>	USART serial bus class . . . . .	29
<a href="#">UsartDebug</a>	Class used for debugging on usart link . . . . .	32



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

work/main.cpp		
Background task file	78	
work/main.h		
Background task header file	81	
work/asw/asw.cpp		
ASW main file	41	
work/asw/asw.h		
ASW main header file	43	
work/asw/debug/debug.cpp		
This file defines classes for log and debug data transmission on USART link	45	
work/asw/debug/debug.h		
Header file for debug and logging functions	46	
work/asw/keepAliveLed/keepAliveLed.cpp		
Definition of function for class keepAliveLed	48	
work/asw/keepAliveLed/keepAliveLed.h		
Class keepAliveLed header file	49	
work/asw/TempSensor/TempSensor.cpp		
Defines function of class TempSensor	52	
work/asw/TempSensor/TempSensor.h		
Class TempSensor header file	53	
work/bsw/bsw.cpp		
BSW main file	55	
work/bsw/bsw.h		
BSW main header file	57	
work/bsw/dht22/dht22.cpp		
This file defines classes for DHT22 driver	60	
work/bsw/dht22/dht22.h		
DHT22 driver header file	61	
work/bsw/dio/dio.cpp		
DIO library	63	
work/bsw/dio/dio.h		
DIO library header file	63	
work/bsw/dio/dio_port_cnf.h		
Digital ports configuration file	67	
work/bsw/dio/dio_reg_atm2560.h		
	69	

work/bsw/timer/timer.cpp	
Defines function for class timer . . . . .	72
work/bsw/timer/timer.h	
Timer class header file . . . . .	73
work/bsw/usart/usart.cpp	
BSW library for USART . . . . .	73
work/bsw/usart/usart.h	
Header file for USART library . . . . .	74
work/lib/operators.cpp	
C++ operators definitions . . . . .	75
work/lib/operators.h	
C++ operators definitions header file . . . . .	77
work/scheduler/scheduler.cpp	
Defines scheduler class . . . . .	83
work/scheduler/scheduler.h	
Scheduler class header file . . . . .	84

## Chapter 3

# Class Documentation

### 3.1 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.*

#### 3.1.1 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.

#### 3.1.2 Constructor & Destructor Documentation

##### 3.1.2.1 dht22()

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

[dht22](#) class constructor

Initializes the class [dht22](#)

#### Returns

Nothing

Definition at line 22 of file dht22.cpp.

### 3.1.3 Member Function Documentation

#### 3.1.3.1 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

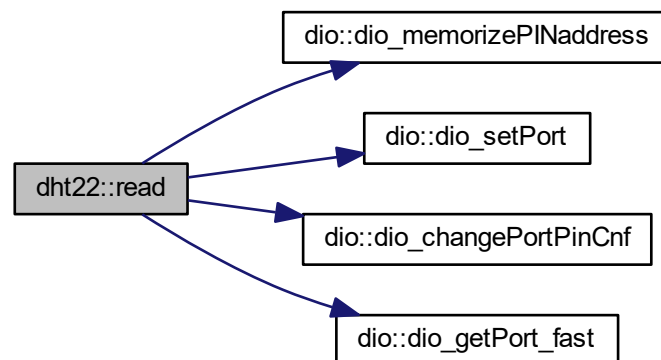
out	<i>raw_humidity</i>	Raw humidity value received from sensor
out	<i>raw_temperature</i>	Raw temperature value received from sensor

#### Returns

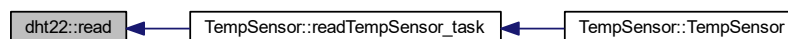
Validity of the read value

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](#)

## 3.2 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.*

### 3.2.1 Detailed Description

DIO class.

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

Definition at line 31 of file dio.h.

### 3.2.2 Constructor & Destructor Documentation

### 3.2.2.1 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.

## 3.2.3 Member Function Documentation

### 3.2.3.1 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:



## 3.2.3.2 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.

## 3.2.3.3 dio\_getPort\_fast()

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

Gets the logical state of the memorized pin.

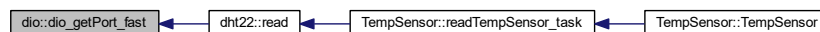
This function gets the logical value of the memorized pin. The corresponding port and pin index are stored in class members PINx\_addr\_mem and PINx\_idx\_mem. This mechanism is used to speed up reading time as this function no longer needs to extract register address and pin index from portcode.

## Returns

Logical state of selected pin

Definition at line 171 of file dio.cpp.

Here is the caller graph for this function:



## 3.2.3.4 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**

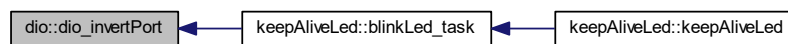
in	<i>portcode</i>	Encoded pin and register index
----	-----------------	--------------------------------

**Returns**

Nothing

Definition at line 131 of file dio.cpp.

Here is the caller graph for this function:

**3.2.3.5 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**

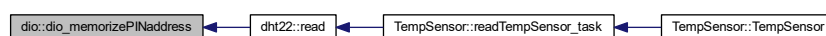
in	<i>portcode</i>	Encoded pin and register index
----	-----------------	--------------------------------

**Returns**

Nothing

Definition at line 165 of file dio.cpp.

Here is the caller graph for this function:



## 3.2.3.6 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](#)

## 3.3 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.*

### 3.3.1 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`.

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 `keepAliveLed()`

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

Class constructor.

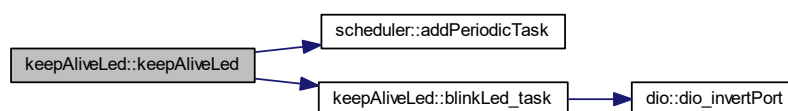
This function initializes the class `keepAliveLed`

#### Returns

Nothing

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

Here is the call graph for this function:



### 3.3.3 Member Function Documentation

### 3.3.3.1 blinkLed\_task()

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

Task for LED blinking.

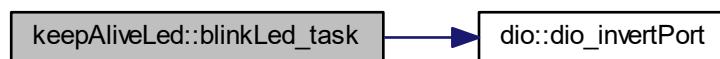
This function is inserted into the scheduler. It changes the state of the LED output to make it blink

#### 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](#)

## 3.4 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.*

### 3.4.1 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.

### 3.4.2 Constructor & Destructor Documentation

#### 3.4.2.1 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:





### 3.4.3 Member Function Documentation

#### 3.4.3.1 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

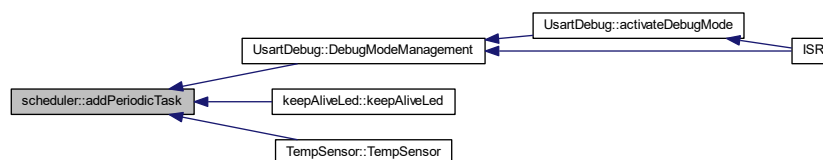
in	<i>task_ptr</i>	Pointer to the task which will be added
in	<i>a_period</i>	Period of the new task

##### Returns

Nothing

Definition at line 63 of file `scheduler.cpp`.

Here is the caller graph for this function:



#### 3.4.3.2 getPitNumber()

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

Get function for PIT number.

This function returns the PIT number

**Returns**

PIT number

Definition at line 93 of file scheduler.cpp.

Here is the caller graph for this function:

**3.4.3.3 launchPeriodicTasks()**

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

Main scheduler function.

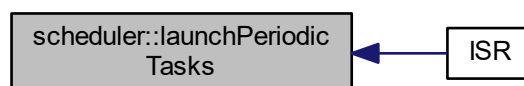
This function launches the scheduled tasks according to current software time and task configuration

**Returns**

Nothing

Definition at line 32 of file scheduler.cpp.

Here is the caller graph for this function:

**3.4.3.4 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 99 of file scheduler.cpp.

Here is the caller graph for this function:



## 3.4.3.5 startScheduling()

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

Starts the tasks scheduling.

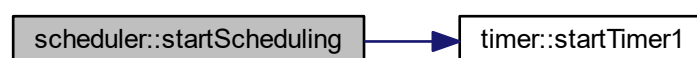
This function starts the timer which will trigger an interrupt every software period. When the interrupt is raised the scheduler will launch applications

## Returns

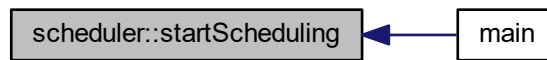
Nothing

Definition at line 57 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:

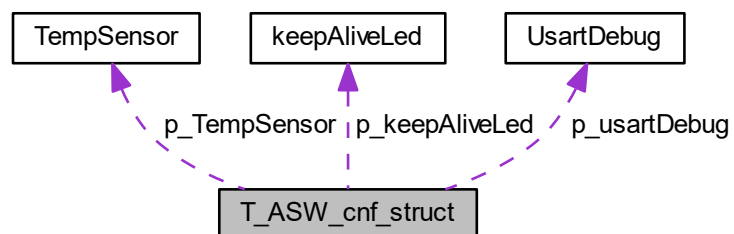
- [work/scheduler/scheduler.h](#)
- [work/scheduler/scheduler.cpp](#)

### 3.5 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`

#### 3.5.1 Detailed Description

ASW configuration structure.

This structure contains all pointers to instanced applicative objects

Definition at line 23 of file `asw.h`.

### 3.5.2 Member Data Documentation

#### 3.5.2.1 p\_keepAliveLed

`keepAliveLed* T_ASW_cnf_struct::p_keepAliveLed`

Pointer to `keepAliveLed` object

Definition at line 26 of file `asw.h`.

#### 3.5.2.2 p\_TempSensor

`TempSensor* T_ASW_cnf_struct::p_TempSensor`

Pointer to `TempSensor` object

Definition at line 27 of file `asw.h`.

#### 3.5.2.3 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:

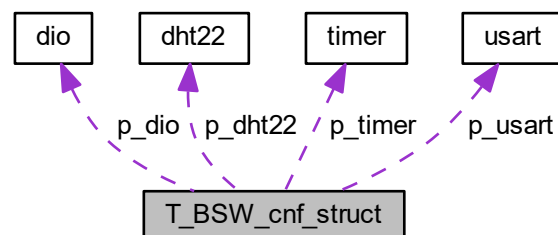
- `work/asw/asw.h`

## 3.6 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](#)

### 3.6.1 Detailed Description

BSW configuration structure.

This structure contains all pointers to instanced drivers objects

Definition at line 29 of file bsw.h.

### 3.6.2 Member Data Documentation

#### 3.6.2.1 p\_dht22

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

Pointer to [dht22](#) driver object

Definition at line 34 of file bsw.h.

#### 3.6.2.2 p\_dio

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

Pointer to dio driver object

Definition at line 32 of file bsw.h.

#### 3.6.2.3 p\_timer

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

Pointer to timer driver object

Definition at line 33 of file bsw.h.

### 3.6.2.4 p\_usart

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

Pointer to usart driver object

Definition at line 31 of file bsw.h.

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

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

## 3.7 TempSensor Class Reference

```
#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.*

### 3.7.1 Detailed Description

Definition at line 15 of file TempSensor.h.

### 3.7.2 Constructor & Destructor Documentation

### 3.7.2.1 TempSensor()

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

Class constructor.

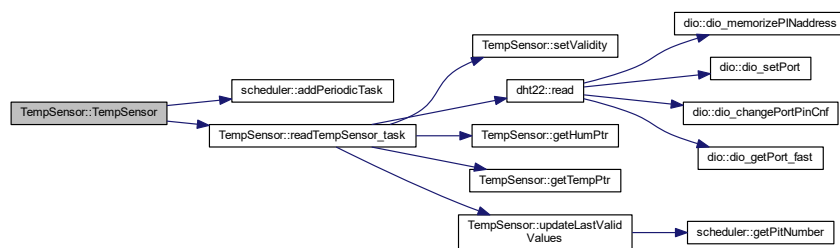
This function initializes all data of the class [TempSensor](#)

#### Returns

Nothing

Definition at line 16 of file TempSensor.cpp.

Here is the call graph for this function:



## 3.7.3 Member Function Documentation

### 3.7.3.1 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	<i>hum</i>	Humidity value
-----	------------	----------------

#### Returns

Validity of humidity

Definition at line 66 of file TempSensor.cpp.



Here is the caller graph for this function:



### 3.7.3.2 getHumPtr()

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

Get pointer to data raw\_humidity.

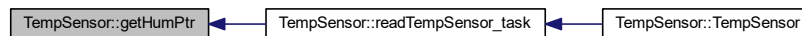
This function returns a pointer to the class member raw\_humidity

#### Returns

Pointer to raw\_humidity

Definition at line 41 of file TempSensor.cpp.

Here is the caller graph for this function:



### 3.7.3.3 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:

**3.7.3.4 getTempPtr()**

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

Get pointer to data raw\_temperature.

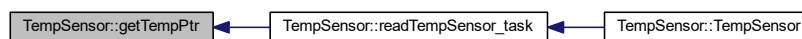
This function returns a pointer to the class member raw\_temperature

**Returns**

Pointer to raw\_temperature

Definition at line 46 of file TempSensor.cpp.

Here is the caller graph for this function:

**3.7.3.5 readTempSensor\_task()**

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

Task for reading temperature and humidity values.

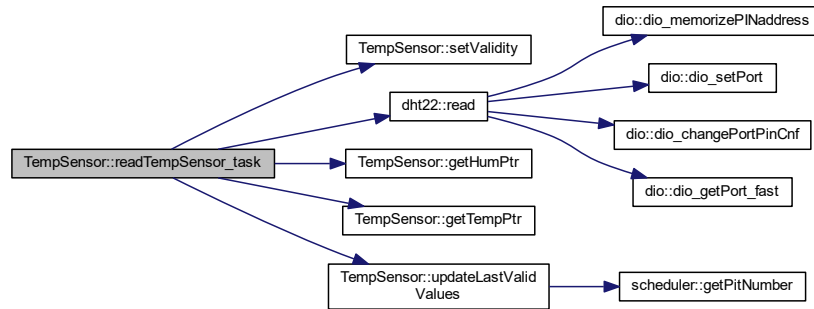
This task reads temperature and humidity data using DHT22 driver. It is called every 5 seconds.

**Returns**

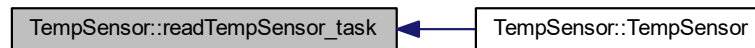
Nothing

Definition at line 30 of file TempSensor.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**3.7.3.6 setValidity()**

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

Set data val\_validity.

This function sets the class member val\_validity

**Parameters**

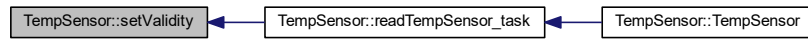
in	<i>validity</i>	Value of validity
----	-----------------	-------------------

**Returns**

Nothing

Definition at line 36 of file TempSensor.cpp.

Here is the caller graph for this function:

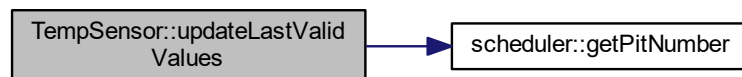


### 3.7.3.7 updateLastValidValues()

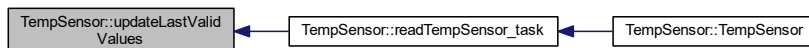
```
void TempSensor::updateLastValidValues ( )
```

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](#)

## 3.8 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.*

### 3.8.1 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.

### 3.8.2 Constructor & Destructor Documentation

#### 3.8.2.1 timer()

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

Class constructor.

This function initializes class attributes

#### Returns

Nothing

Definition at line 13 of file timer.cpp.

### 3.8.3 Member Function Documentation

#### 3.8.3.1 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:

**3.8.3.2 startTimer1()**

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

Start Timer #1.

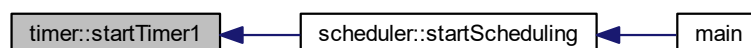
This functions starts Timer #1. Timer shall be initialized before this function is called.

**Returns**

Nothing

Definition at line 56 of file timer.cpp.

Here is the caller graph for this function:



### 3.8.3.3 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

## 3.9 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.*

### 3.9.1 Detailed Description

USART serial bus class.

This class defines all useful functions for USART serial bus

Definition at line 16 of file usart.h.

### 3.9.2 Constructor & Destructor Documentation

#### 3.9.2.1 usart()

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

Class usart constructor.

Initializes the class and call hardware initialization function

**Parameters**

in	<i>a_BaudRate</i>	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:



### 3.9.3 Member Function Documentation

#### 3.9.3.1 `setBaudRate()`

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

Setting baud rate.

This function sets the attribute `BaudRate` of the class `usart`

**Parameters**

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

**Returns**

Nothing

Definition at line 63 of file `usart.cpp`.



### 3.9.3.2 usart\_init()

```
void usart::usart_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:



### 3.9.3.3 usart\_read()

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

USART read function.

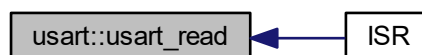
This function will read reception register of USART

#### 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:



### 3.9.3.4 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

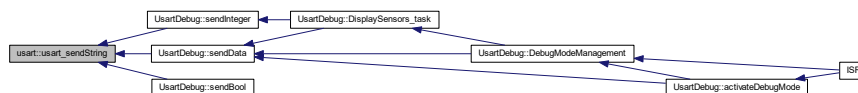
in	str	Pointer to the string being sent
----	-----	----------------------------------

#### 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](#)

## 3.10 UsartDebug Class Reference

Class used for debugging on usart link.

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

### Public Member Functions

- [UsartDebug](#) ()  
*Class [UsartDebug](#) constructor.*
- void [sendData](#) (char \*str)  
*Send a string on USART link.*
- 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.*

### 3.10.1 Detailed Description

Class used for debugging on usart link.

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

Definition at line 31 of file debug.h.

### 3.10.2 Constructor & Destructor Documentation

#### 3.10.2.1 UsartDebug()

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

Class [UsartDebug](#) constructor.

Initializes the class [UsartDebug](#)

#### Returns

Nothing

Definition at line 30 of file debug.cpp.

### 3.10.3 Member Function Documentation

### 3.10.3.1 activateDebugMode()

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

Activates debug mode.

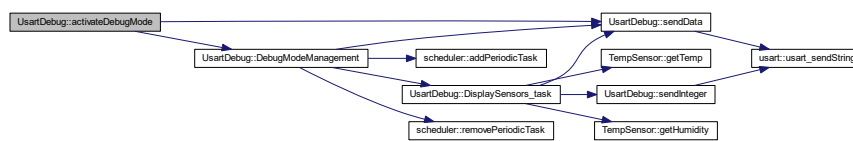
This function activates USART debug mode.

#### Returns

Nothing

Definition at line 114 of file debug.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 3.10.3.2 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

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

## Parameters

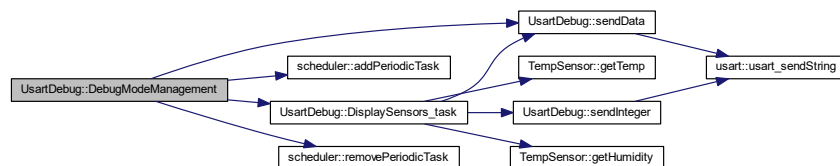
in	<i>rcv_char</i>	8 bits character received on USART
----	-----------------	------------------------------------

## Returns

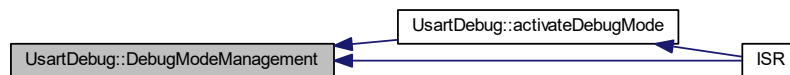
Nothing

Definition at line 122 of file debug.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 3.10.3.3 DisplaySensors\_task()

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

Displays sensors data on usart link.

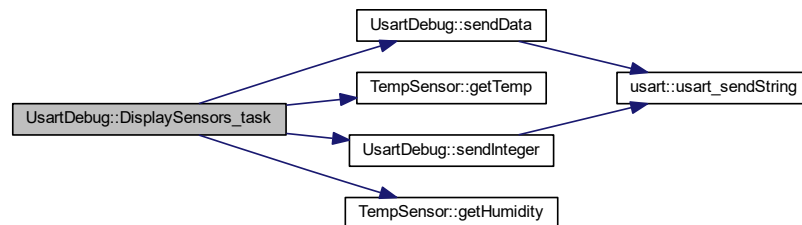
This task sends sensors data (temperature and humidity) on usart link every 5 seconds

**Returns**

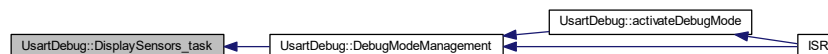
Nothing

Definition at line 68 of file debug.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**3.10.3.4 isDebugModeActive()**

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

Check is debug mode is active or not.

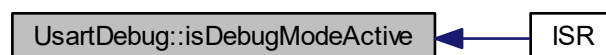
This function checks if debug mode is active or not.

**Returns**

TRUE is debug mode is active, FALSE otherwise

Definition at line 109 of file debug.cpp.

Here is the caller graph for this function:



### 3.10.3.5 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 56 of file debug.cpp.

Here is the call graph for this function:



### 3.10.3.6 sendData()

```
void UsartDebug::sendData (
    char * str )
```

Send a string on USART link.

This functions sends the requested string on USART link by calling driver's transmission function

#### Parameters

in	<i>str</i>	Pointer to the string being sent
----	------------	----------------------------------

#### Returns

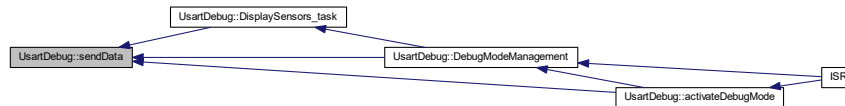
Nothing

Definition at line 36 of file debug.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 3.10.3.7 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 42 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

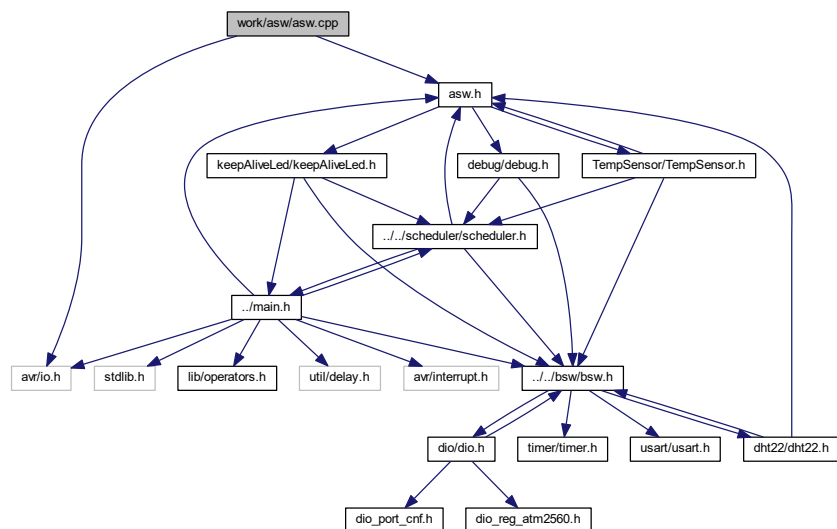
# File Documentation

### 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

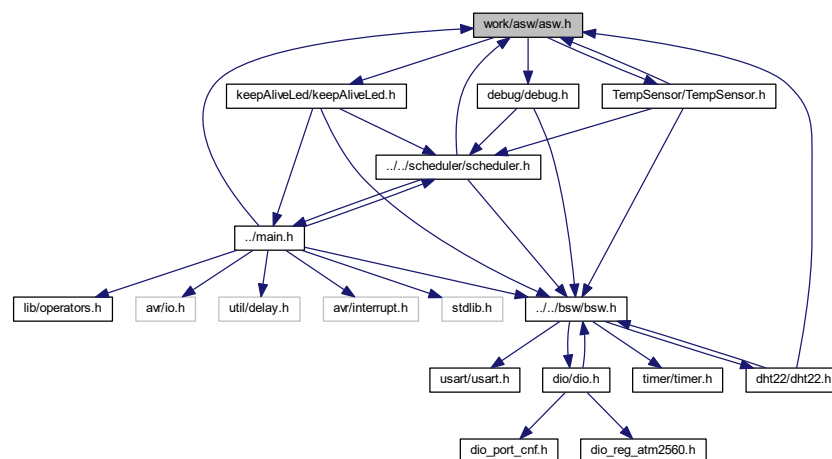
ASW configuration structure

Definition at line 17 of file asw.cpp.

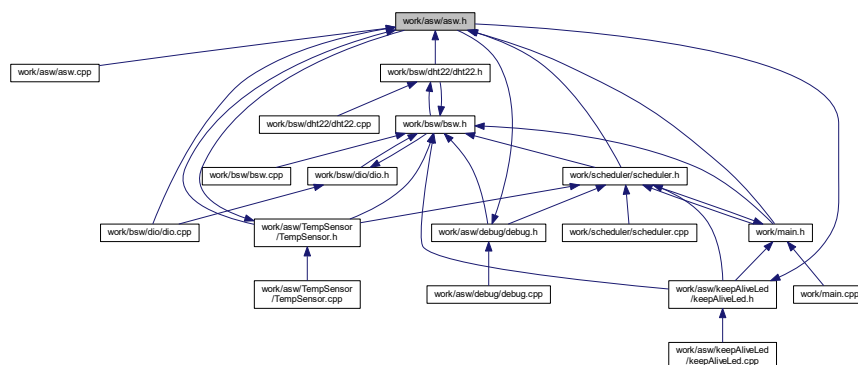
## 4.2 work/asw/asw.h File Reference

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`

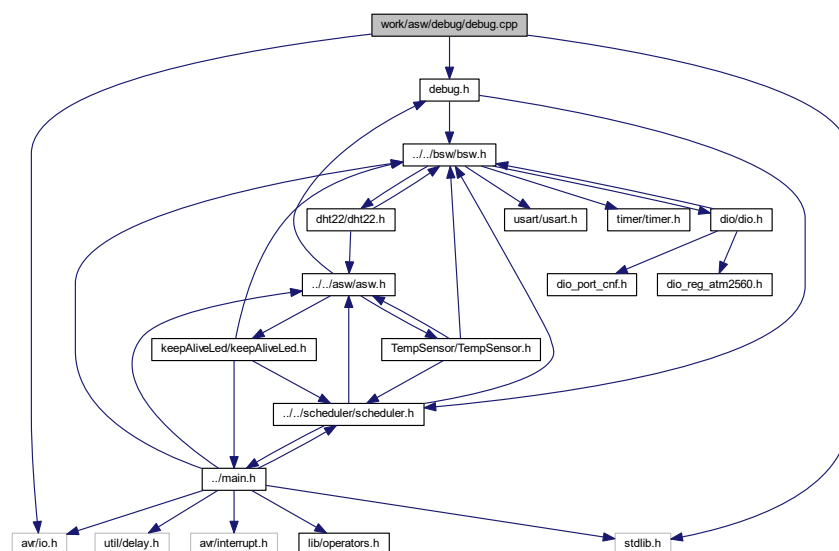
ASW configuration structure

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"
    "\n"
    "s : Quitter debug\n"
```

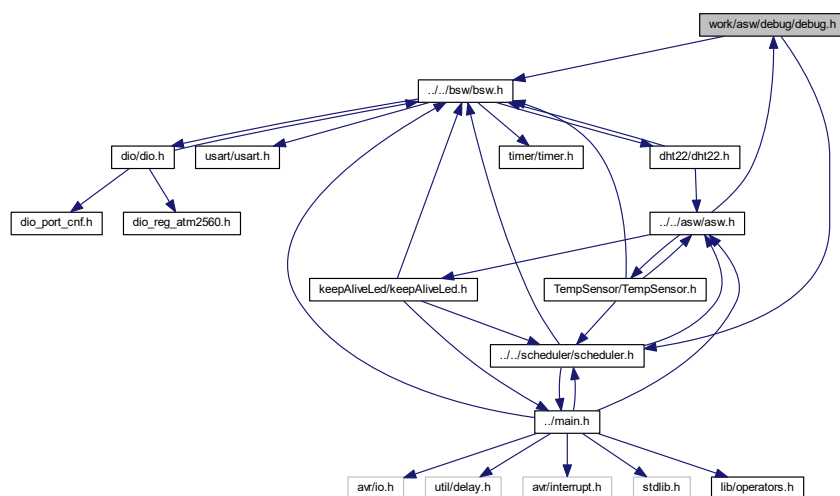
Main menu of debug mode.

Definition at line 20 of file debug.cpp.

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

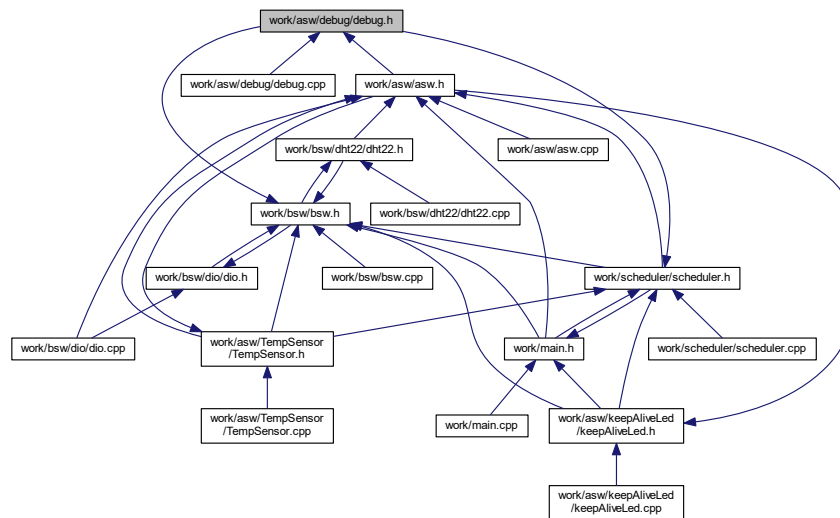
Header file for debug and logging functions.

```
#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

## Enumerations

- enum [debug\\_state\\_t](#) { INIT, WAIT\_INIT, DISPLAY\_DATA }  
*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\_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

Definition at line 19 of file debug.h.

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

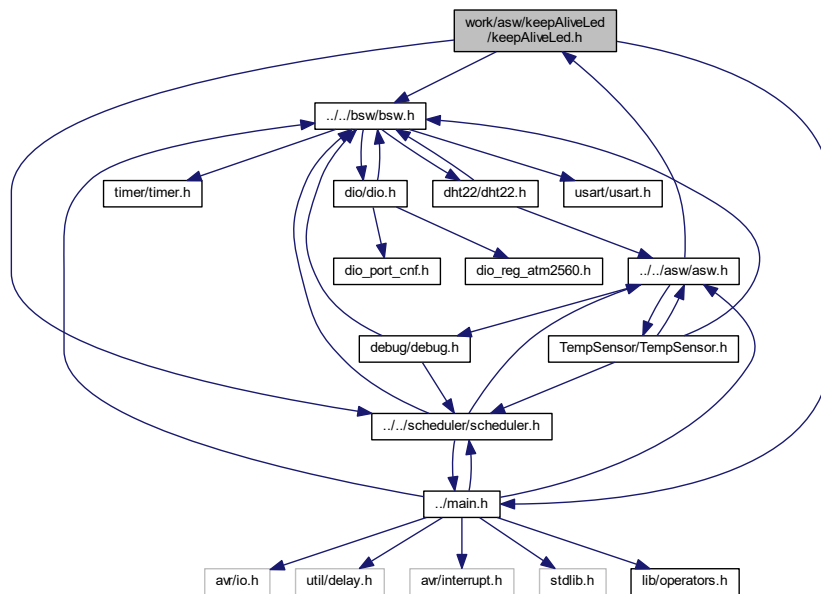
Definition of function for class [keepAliveLed](#).

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

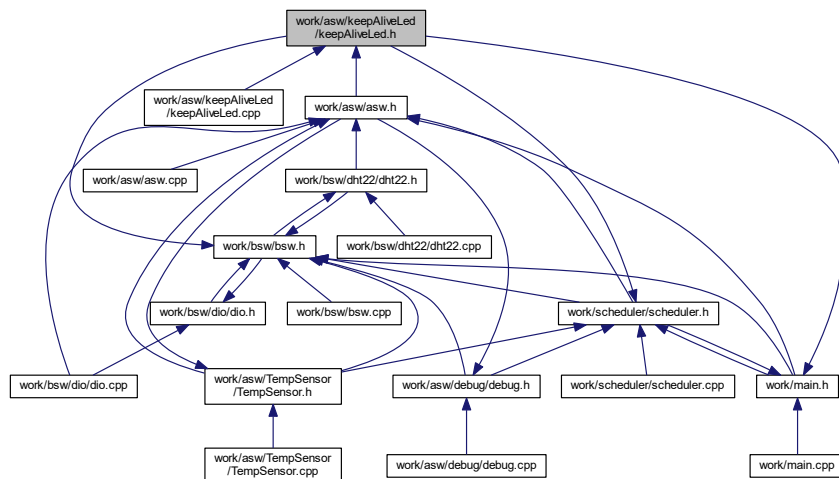


```
#include "../../main.h"
```

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
```

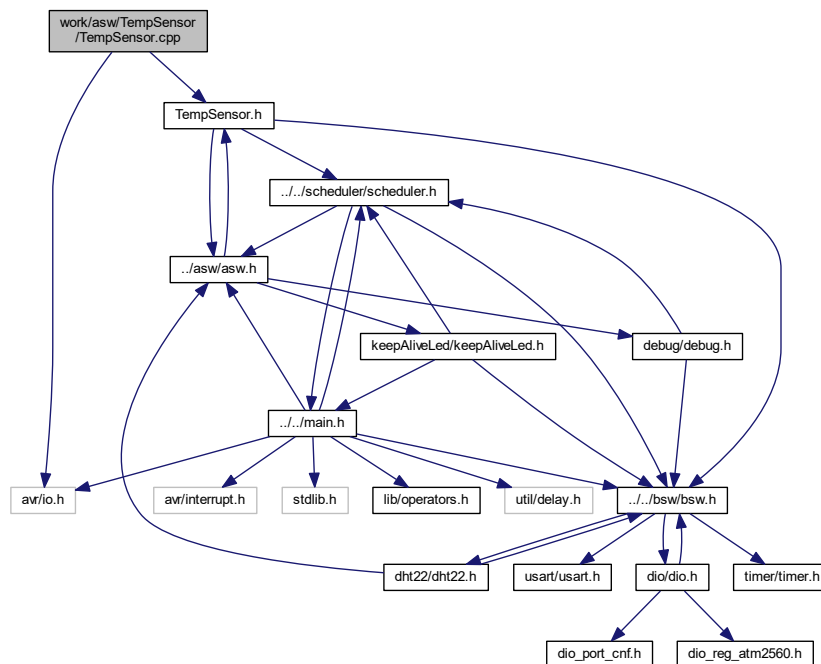
Period for led blinking

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
```

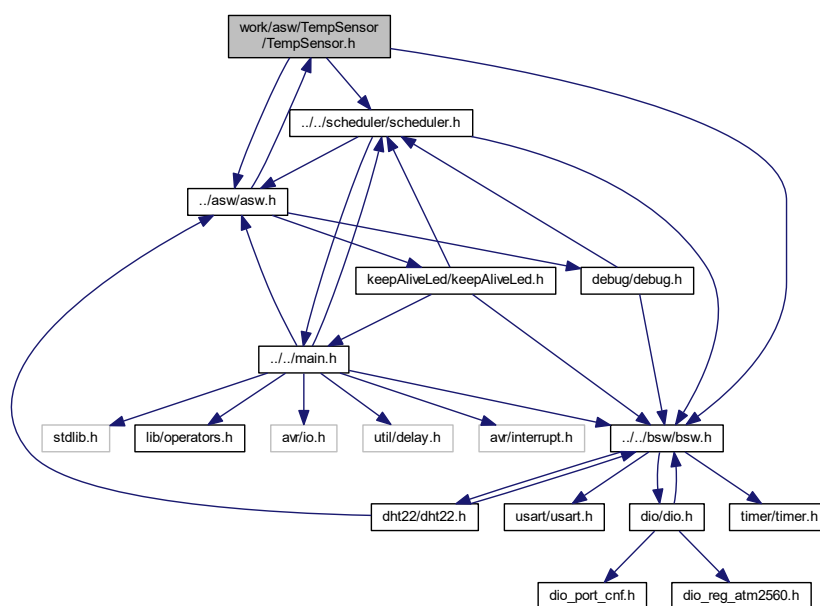
Definition at line 14 of file TempSensor.cpp.

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

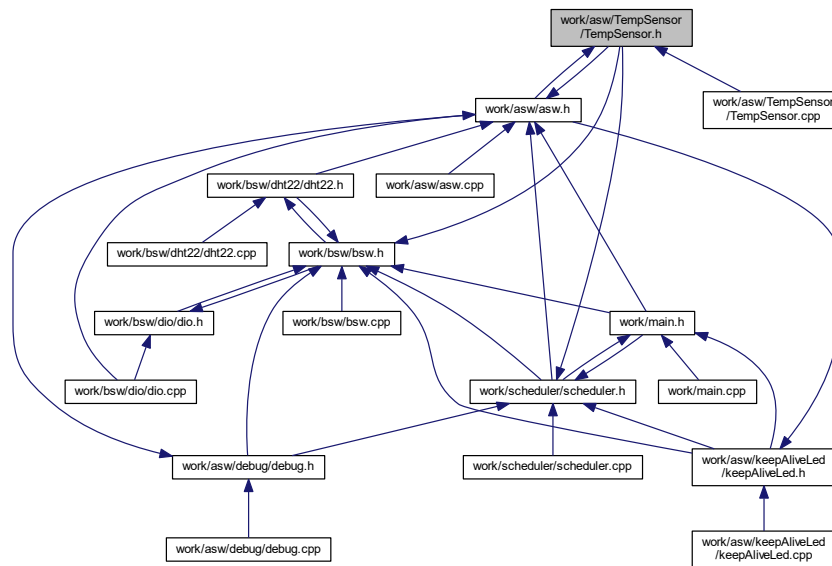
Class [TempSensor](#) header file.

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

Include dependency graph for TempSensor.h:



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



## Classes

- class [TempSensor](#)

## Macros

- `#define` [PERIOD\\_MS\\_TASK\\_TEMP\\_SENSOR](#) 5000

### 4.8.1 Detailed Description

Class [TempSensor](#) header file.

#### Date

23 mars 2018

#### Author

nicls67

### 4.8.2 Macro Definition Documentation



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

```
#define PERIOD_MS_TASK_TEMP_SENSOR 5000
```

Period for reading temperature data

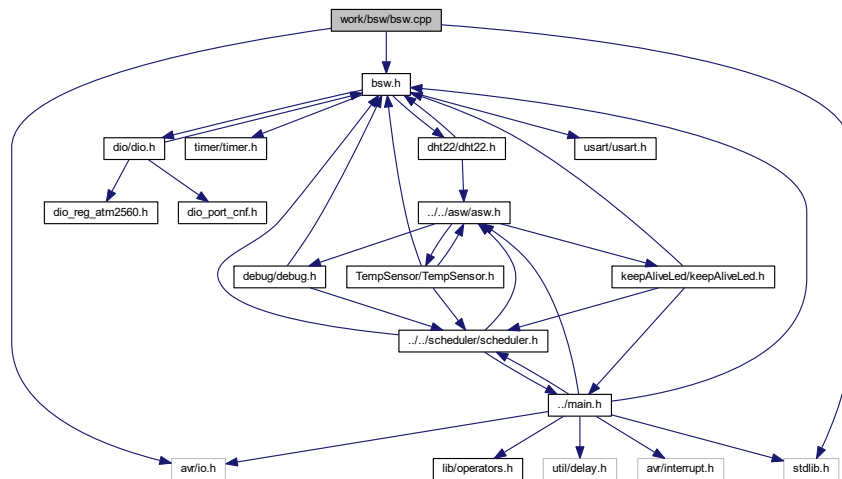
Definition at line 13 of file TempSensor.h.

## 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

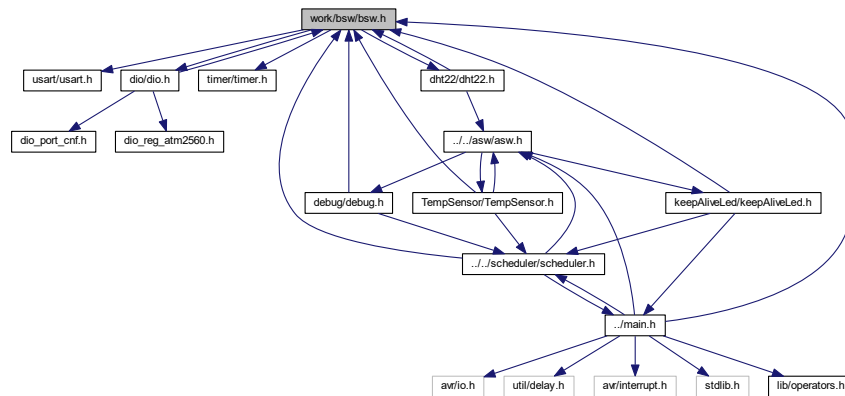
Definition at line 16 of file bsw.cpp.

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

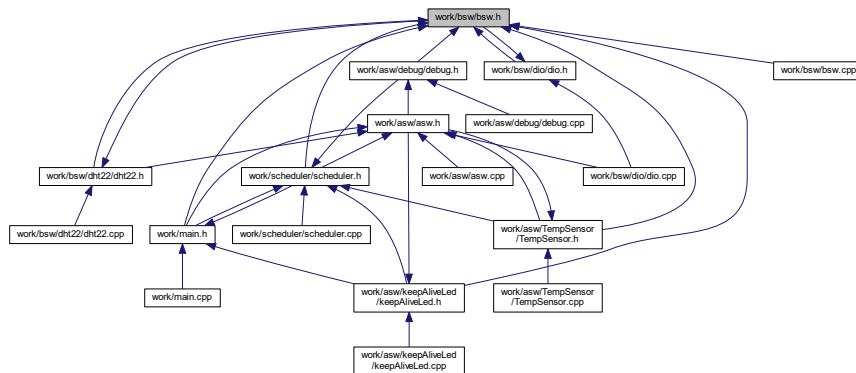
BSW main header file.

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

Include 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 23 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

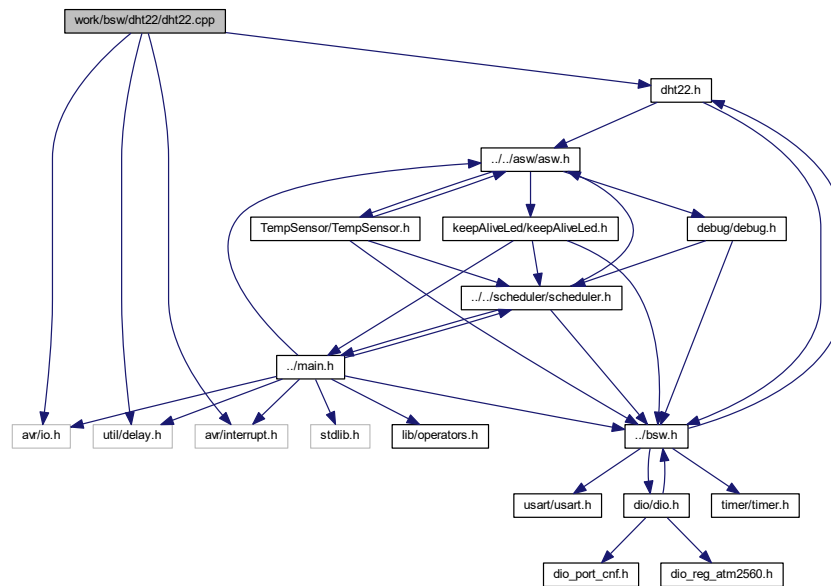
Definition at line 16 of file bsw.cpp.

## 4.11 work/bsw/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

#### 4.11.1 Detailed Description

This file defines classes for DHT22 driver.

#### Date

23 mars 2018

#### Author

nicls67

#### 4.11.2 Macro Definition Documentation

## 4.11.2.1 MAX\_WAIT\_TIME\_US

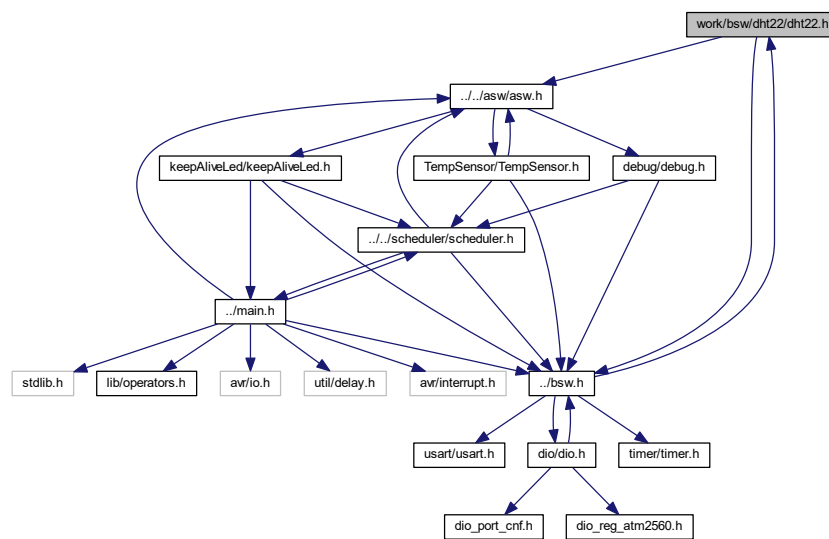
```
#define MAX_WAIT_TIME_US 100
```

Definition at line 20 of file dht22.cpp.

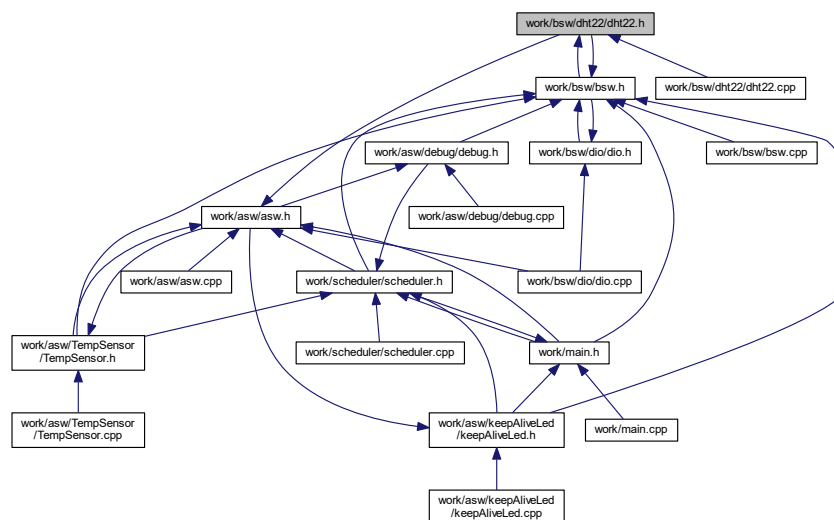
## 4.12 work/bsw/dht22/dht22.h File Reference

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)`

### 4.12.1 Detailed Description

DHT22 driver header file.

#### Date

23 mars 2018

#### Author

nicls67

### 4.12.2 Macro Definition Documentation

#### 4.12.2.1 DHT22\_PORT

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

DHT22 is connected to port PB6

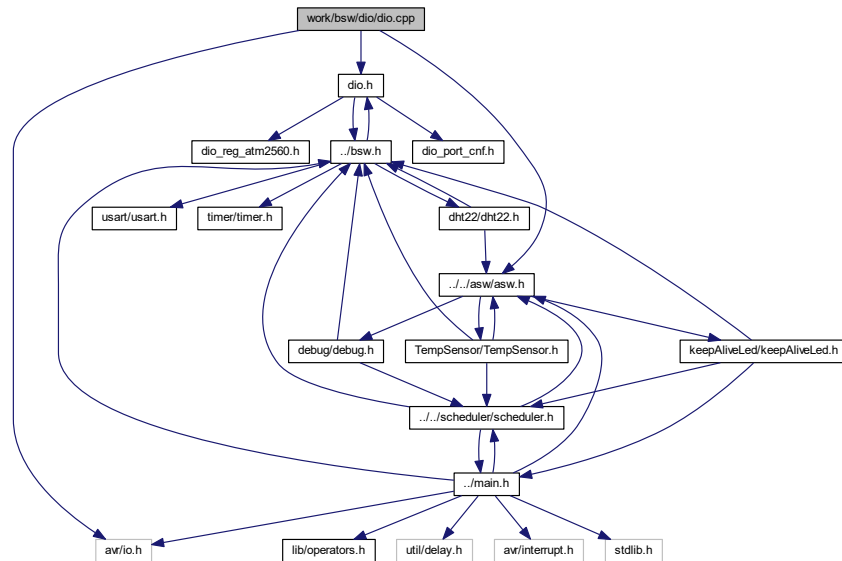
Definition at line 16 of file dht22.h.



## 4.13 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:
```



### 4.13.1 Detailed Description

DIO library.

Date

13 mars 2018

Author

nicls67

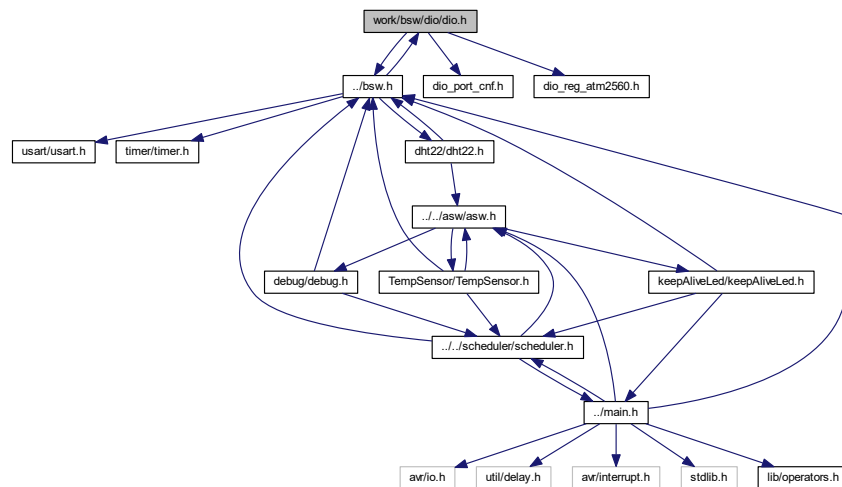
## 4.14 work/bsw/dio/dio.h File Reference

DIO library header file.

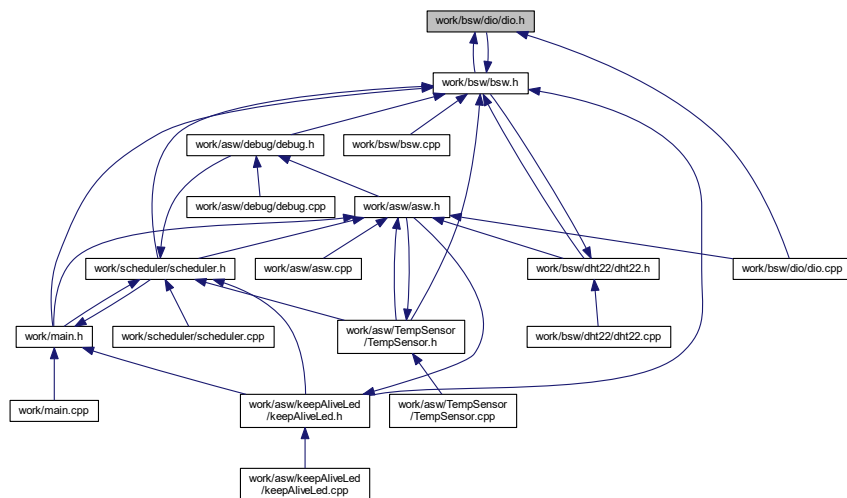
```
#include "../bsw.h"
#include "dio_port_cnf.h"
```

```
#include "dio_reg_atm2560.h"
```

Include dependency graph for dio.h:



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



## Classes

- class [dio](#)  
*DIO class.*

## Macros

- `#define PORT_CONF_OUT 1`
- `#define PORT_CONF_IN 0`

- #define `ENCODE_PORT`(port, pin) (uint8\_t)((((uint8\_t)(port & 0xF)) << 3) | (uint8\_t)(pin & 0x7))
- #define `DECODE_PORT`(portcode) (uint8\_t)((portcode >> 3) & 0xF)
- #define `DECODE_PIN`(portcode) (uint8\_t)(portcode & 0x7)
- #define `PORT_A` 0
- #define `PORT_B` 1
- #define `PORT_C` 2
- #define `PORT_D` 3

### 4.14.1 Detailed Description

DIO library header file.

Date

13 mars 2018

Author

nicls67

### 4.14.2 Macro Definition Documentation

#### 4.14.2.1 DECODE\_PIN

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

Macro used to extract pin index

Definition at line 19 of file dio.h.

#### 4.14.2.2 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.

#### 4.14.2.3 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.

#### 4.14.2.4 PORT\_A

```
#define PORT_A 0
```

PORTA index

Definition at line 21 of file dio.h.

#### 4.14.2.5 PORT\_B

```
#define PORT_B 1
```

PORTB index

Definition at line 22 of file dio.h.

#### 4.14.2.6 PORT\_C

```
#define PORT_C 2
```

PORTC index

Definition at line 23 of file dio.h.

#### 4.14.2.7 PORT\_CNF\_IN

```
#define PORT_CNF_IN 0
```

Pin is configured as input

Definition at line 15 of file dio.h.

## 4.14.2.8 PORT\_CNF\_OUT

```
#define PORT_CNF_OUT 1
```

Pin is configured as output

Definition at line 14 of file dio.h.

## 4.14.2.9 PORT\_D

```
#define PORT_D 3
```

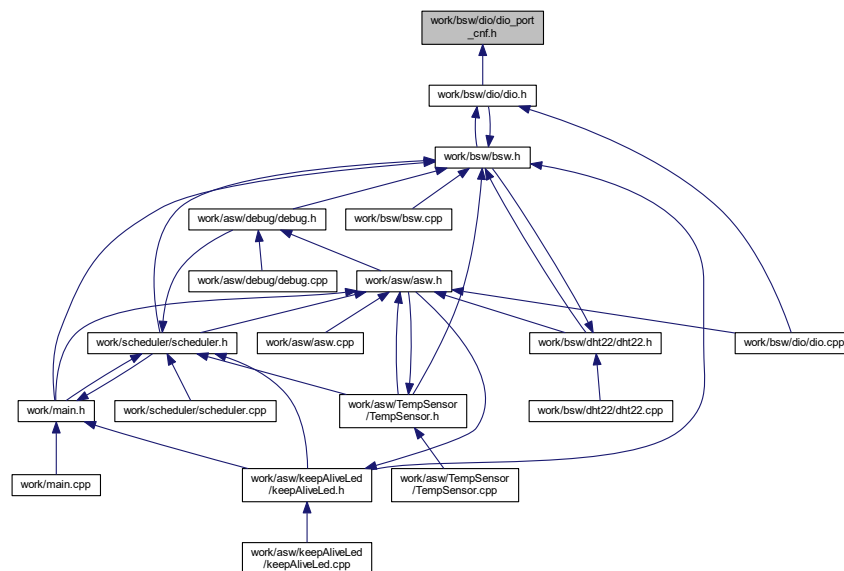
PORTD index

Definition at line 24 of file dio.h.

## 4.15 work/bsw/dio/dio\_port\_conf.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.

### 4.15.1 Detailed Description

Digital ports configuration file.

#### Date

19 mars 2019

#### Author

nicls67

### 4.15.2 Macro Definition Documentation

#### 4.15.2.1 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.

#### 4.15.2.2 PORTB\_CNF\_PORTB

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

Defines the configuration of PORTB register.

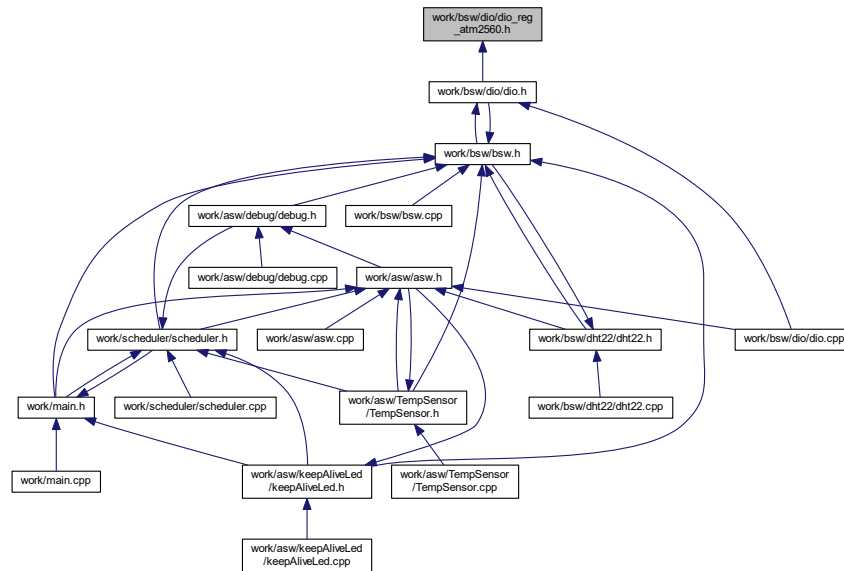
This constant defines the initial value of IO pins for PORT B. It will configure register PORTB. Pins configured as input shall not be configured here.

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.

## 4.16 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)`

### 4.16.1 Macro Definition Documentation

#### 4.16.1.1 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.

#### 4.16.1.2 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.

#### 4.16.1.3 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.

#### 4.16.1.4 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.

#### 4.16.1.5 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.

#### 4.16.1.6 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.



#### 4.16.1.7 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.

#### 4.16.1.8 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.

#### 4.16.1.9 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.

#### 4.16.1.10 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.

#### 4.16.1.11 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.

#### 4.16.1.12 PORTD\_PTR

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

Macro defining pointer to PORT D register

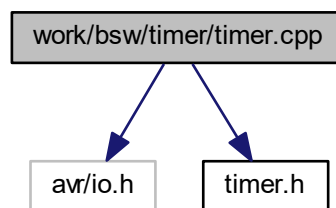
Definition at line 17 of file dio\_reg\_atm2560.h.

### 4.17 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:



#### 4.17.1 Detailed Description

Defines function for class timer.

Date

15 mars 2018

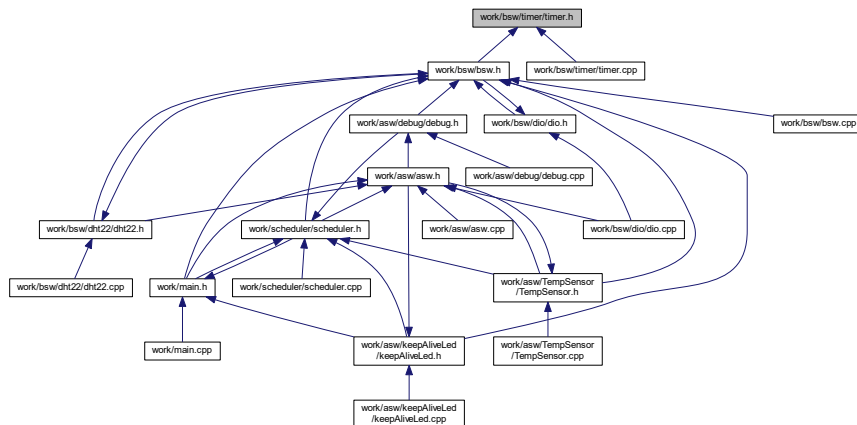
Author

nicls67

## 4.18 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.*

### 4.18.1 Detailed Description

Timer class header file.

#### Date

15 mars 2018

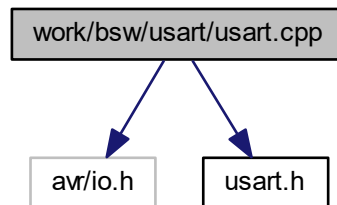
#### Author

nicls67

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

BSW library for USART.

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



#### 4.19.1 Detailed Description

BSW library for USART.

Date

13 mars 2018

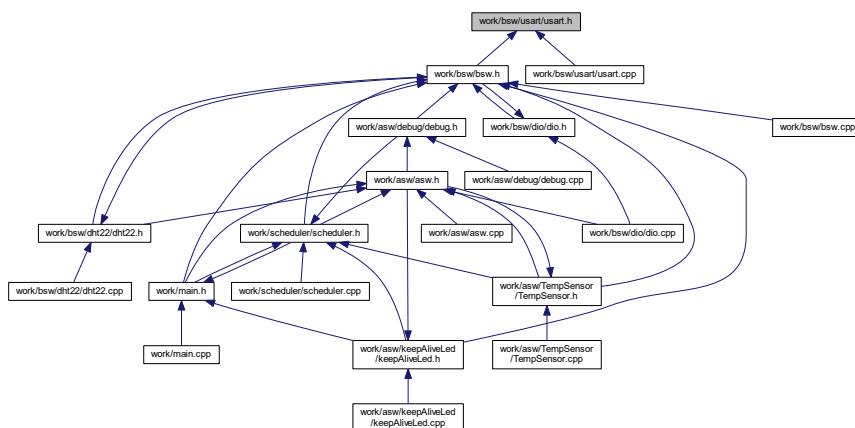
Author

nicls67

#### 4.20 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.*

### 4.20.1 Detailed Description

Header file for USART library.

#### Date

13 mars 2018

#### Author

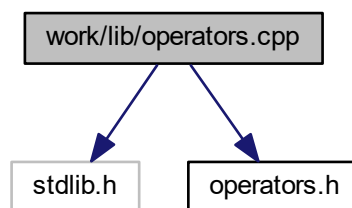
nicls67

## 4.21 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.*

### 4.21.1 Detailed Description

c++ operators definitions

Date

14 mars 2018

Author

nicls67

### 4.21.2 Function Documentation

#### 4.21.2.1 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.

#### 4.21.2.2 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**

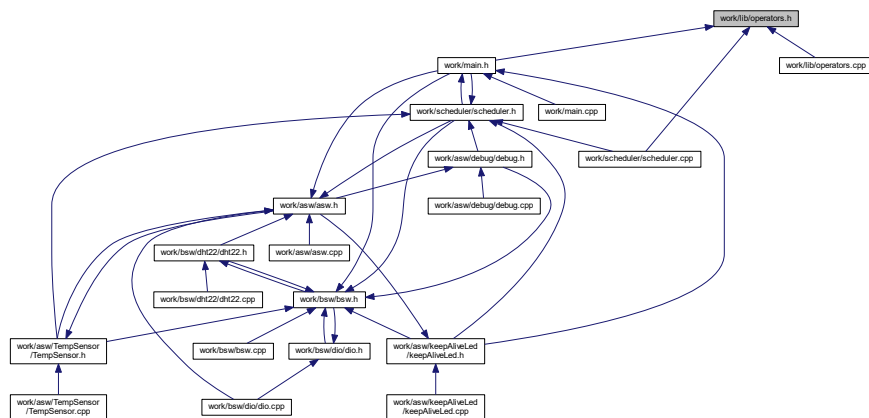
Pointer to the start of allocated memory zone

Definition at line 13 of file operators.cpp.

## 4.22 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.*

### 4.22.1 Detailed Description

c++ operators definitions header file

**Date**

14 mars 2018

**Author**

nicls67

### 4.22.2 Function Documentation

#### 4.22.2.1 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.

#### 4.22.2.2 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.

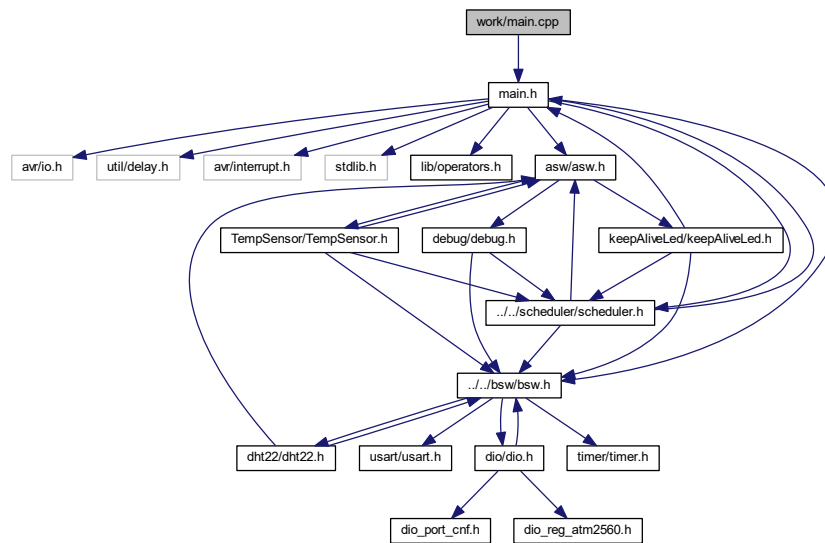
## 4.23 work/main.cpp File Reference

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.*

### 4.23.1 Detailed Description

Background task file.

Date

12 mars 2018

Author

nicls67

### 4.23.2 Function Documentation

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

```
ISR (
    TIMER1_COMPA_vect )
```

Main software interrupt.

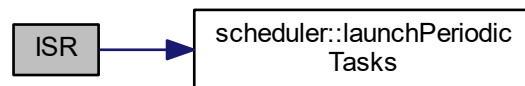
This function handles the interrupt raised by Timer #1. It wakes up the software every 500 ms to perform applications.

##### Returns

Nothing

Definition at line 19 of file main.cpp.

Here is the call graph for this function:



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

```
ISR (
    USART0_RX_vect )
```

USART Rx Complete interrupt.

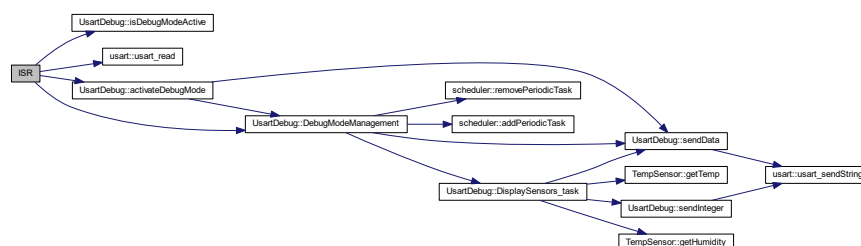
This function handles the interrupt raised when a frame has been received by USART. If debug mode mode is active, it calls debug mode management function. If inactive, it calls debug mode activation function if the received character is 'a'

##### Returns

Nothing

Definition at line 31 of file main.cpp.

Here is the call graph for this function:



## 4.23.2.3 main()

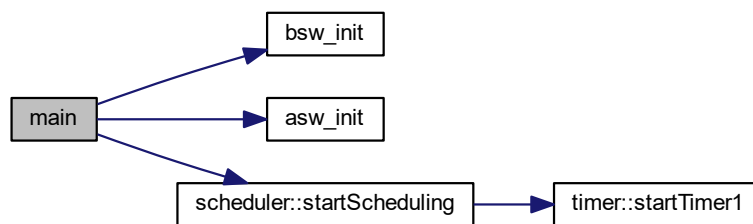
```
int main (  
    void )
```

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:

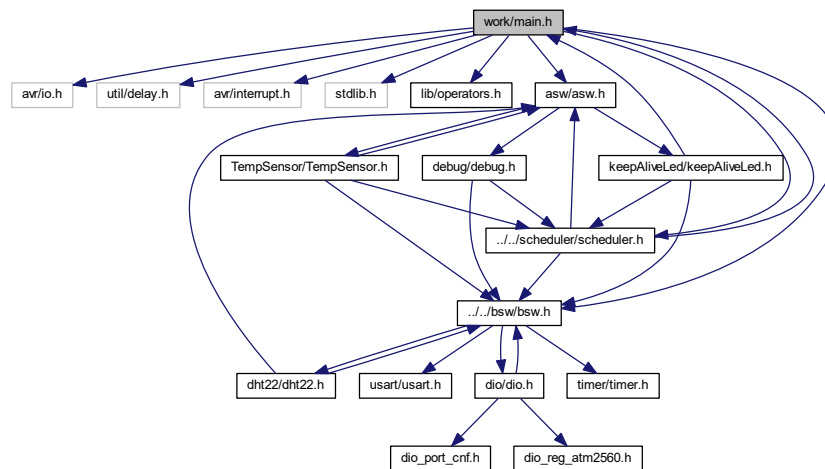


## 4.24 work/main.h File Reference

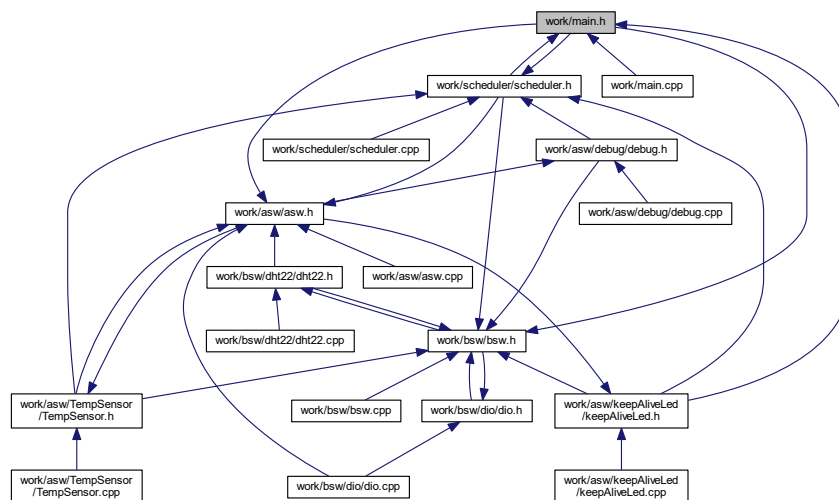
Background task header file.

```
#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:



#### 4.24.1 Detailed Description

Background task header file.

Date

17 mars 2018

Author

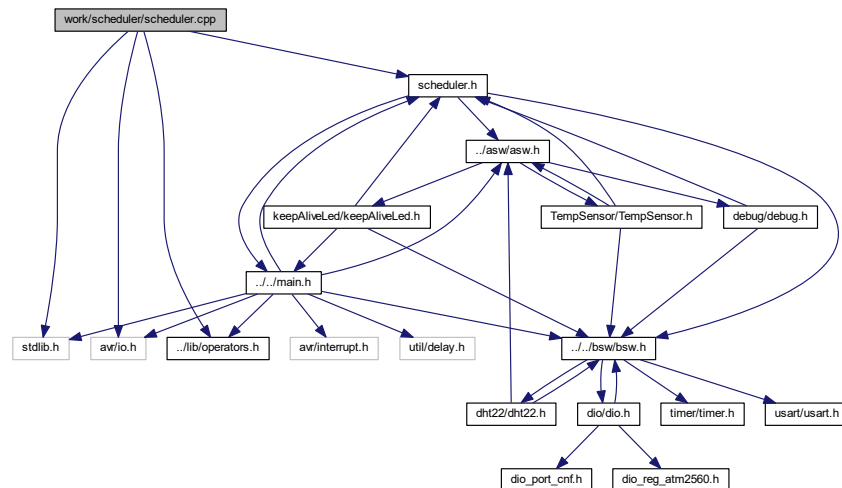
nicls67

## 4.25 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`

### 4.25.1 Detailed Description

Defines scheduler class.

Date

16 mars 2018

Author

nicls67

### 4.25.2 Variable Documentation

#### 4.25.2.1 p\_scheduler

```
scheduler* p_scheduler
```

Pointer to scheduler object

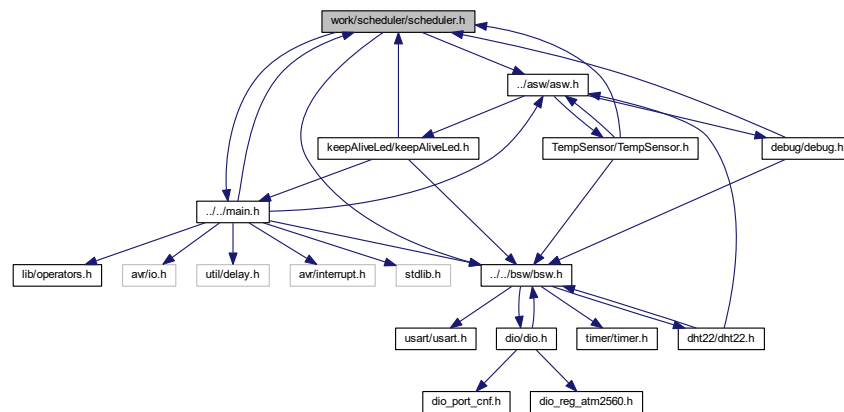
Definition at line 17 of file scheduler.cpp.

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

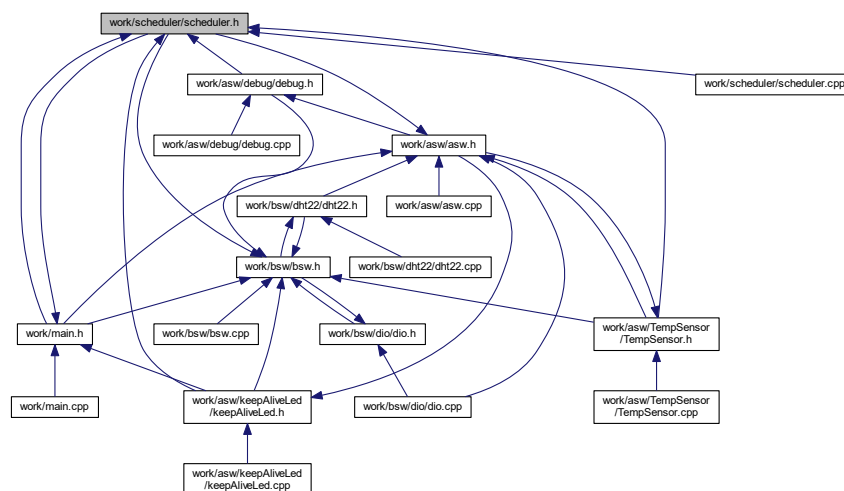
Scheduler class header file.

```
#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`

### 4.26.1 Detailed Description

Scheduler class header file.

#### Date

16 mars 2018

#### Author

nicls67

### 4.26.2 Macro Definition Documentation

#### 4.26.2.1 PRESCALER\_PERIODIC\_TIMER

```
#define PRESCALER_PERIODIC_TIMER 256
```

Value of prescaler to use for periodic timer

Definition at line 19 of file scheduler.h.

#### 4.26.2.2 SW\_PERIOD\_MS

```
#define SW_PERIOD_MS 500
```

Software period, used to define periodic timer interrupt

Definition at line 18 of file scheduler.h.

#### 4.26.2.3 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.

### 4.26.3 Typedef Documentation

#### 4.26.3.1 TaskPtr\_t

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

Type defining a pointer to function.

Definition at line 25 of file scheduler.h.

### 4.26.4 Variable Documentation

#### 4.26.4.1 p\_scheduler

```
scheduler* p_scheduler
```

Pointer to scheduler object

Definition at line 17 of file scheduler.cpp.



# Index

- ASW\_cnf\_struct
  - asw.cpp, [42](#)
  - asw.h, [45](#)
- activateDebugMode
  - UsartDebug, [33](#)
- addPeriodicTask
  - scheduler, [15](#)
- asw.cpp
  - ASW\_cnf\_struct, [42](#)
  - asw\_init, [42](#)
- asw.h
  - ASW\_cnf\_struct, [45](#)
  - asw\_init, [44](#)
- asw\_init
  - asw.cpp, [42](#)
  - asw.h, [44](#)
- BSW\_cnf\_struct
  - bsw.cpp, [56](#)
  - bsw.h, [59](#)
- blinkLed\_task
  - keepAliveLed, [12](#)
- bsw.cpp
  - BSW\_cnf\_struct, [56](#)
  - bsw\_init, [56](#)
- bsw.h
  - BSW\_cnf\_struct, [59](#)
  - bsw\_init, [58](#)
  - USART\_BAUDRATE, [58](#)
- bsw\_init
  - bsw.cpp, [56](#)
  - bsw.h, [58](#)
- configureTimer1
  - timer, [27](#)
- DDRA\_PTR
  - dio\_reg\_atm2560.h, [69](#)
- DDRB\_PTR
  - dio\_reg\_atm2560.h, [69](#)
- DDRC\_PTR
  - dio\_reg\_atm2560.h, [70](#)
- DDRD\_PTR
  - dio\_reg\_atm2560.h, [70](#)
- DECODE\_PIN
  - dio.h, [65](#)
- DECODE\_PORT
  - dio.h, [65](#)
- DHT22\_PORT
  - dht22.h, [62](#)
- debug.cpp
  - str\_debug\_main\_menu, [46](#)
- debug.h
  - debug\_state\_t, [48](#)
  - PERIOD\_MS\_TASK\_DISPLAY\_SENSORS, [48](#)
- debug\_state\_t
  - debug.h, [48](#)
- DebugModeManagement
  - UsartDebug, [34](#)
- dht22, [5](#)
  - dht22, [5](#)
  - read, [6](#)
- dht22.cpp
  - MAX\_WAIT\_TIME\_US, [60](#)
- dht22.h
  - DHT22\_PORT, [62](#)
- dio, [7](#)
  - dio, [7](#)
  - dio\_changePortPinCnf, [8](#)
  - dio\_getPort, [8](#)
  - dio\_getPort\_fast, [9](#)
  - dio\_invertPort, [9](#)
  - dio\_memorizePINaddress, [10](#)
  - dio\_setPort, [10](#)
- dio.h
  - DECODE\_PIN, [65](#)
  - DECODE\_PORT, [65](#)
  - ENCODE\_PORT, [65](#)
  - PORT\_CNF\_IN, [66](#)
  - PORT\_CNF\_OUT, [66](#)
  - PORT\_A, [66](#)
  - PORT\_B, [66](#)
  - PORT\_C, [66](#)
  - PORT\_D, [67](#)
- dio\_changePortPinCnf
  - dio, [8](#)
- dio\_getPort
  - dio, [8](#)
- dio\_getPort\_fast
  - dio, [9](#)
- dio\_invertPort
  - dio, [9](#)
- dio\_memorizePINaddress
  - dio, [10](#)
- dio\_port\_cnf.h
  - PORTB\_CNF\_DDRB, [68](#)
  - PORTB\_CNF\_PORTB, [68](#)
- dio\_reg\_atm2560.h
  - DDRA\_PTR, [69](#)

- DDRB\_PTR, 69
- DDRC\_PTR, 70
- DDRD\_PTR, 70
- PINA\_PTR, 70
- PINB\_PTR, 70
- PINC\_PTR, 70
- PIND\_PTR, 71
- PORTA\_PTR, 71
- PORTB\_PTR, 71
- PORTC\_PTR, 71
- PORTD\_PTR, 71
- dio\_setPort
  - dio, 10
- DisplaySensors\_task
  - UsartDebug, 35
- ENCODE\_PORT
  - dio.h, 65
- getHumPtr
  - TempSensor, 23
- getHumidity
  - TempSensor, 22
- getPitNumber
  - scheduler, 15
- getTemp
  - TempSensor, 23
- getTempPtr
  - TempSensor, 24
- ISR
  - main.cpp, 79, 80
- isDebugModeActive
  - UsartDebug, 36
- keepAliveLed, 11
  - blinkLed\_task, 12
  - keepAliveLed, 12
- keepAliveLed.h
  - LED\_PORT, 51
  - PERIOD\_MS\_TASK\_LED, 51
- LED\_PORT
  - keepAliveLed.h, 51
- launchPeriodicTasks
  - scheduler, 16
- MAX\_WAIT\_TIME\_US
  - dht22.cpp, 60
- main
  - main.cpp, 80
- main.cpp
  - ISR, 79, 80
  - main, 80
- operator delete
  - operators.cpp, 76
  - operators.h, 77
- operator new
  - operators.cpp, 76
- operators.h, 78
- operators.cpp
  - operator delete, 76
  - operator new, 76
- operators.h
  - operator delete, 77
  - operator new, 78
- p\_TempSensor
  - T\_ASW\_cnf\_struct, 19
- p\_dht22
  - T\_BSW\_cnf\_struct, 20
- p\_dio
  - T\_BSW\_cnf\_struct, 20
- p\_keepAliveLed
  - T\_ASW\_cnf\_struct, 19
- p\_scheduler
  - scheduler.cpp, 83
  - scheduler.h, 86
- p\_timer
  - T\_BSW\_cnf\_struct, 20
- p\_usart
  - T\_BSW\_cnf\_struct, 20
- p\_usartDebug
  - T\_ASW\_cnf\_struct, 19
- PERIOD\_MS\_TASK\_DISPLAY\_SENSORS
  - debug.h, 48
- PERIOD\_MS\_TASK\_LED
  - keepAliveLed.h, 51
- PERIOD\_MS\_TASK\_TEMP\_SENSOR
  - TempSensor.h, 54
- PINA\_PTR
  - dio\_reg\_atm2560.h, 70
- PINB\_PTR
  - dio\_reg\_atm2560.h, 70
- PINC\_PTR
  - dio\_reg\_atm2560.h, 70
- PIND\_PTR
  - dio\_reg\_atm2560.h, 71
- PIT\_BEFORE\_INVALID
  - TempSensor.cpp, 52
- PORT\_CNF\_IN
  - dio.h, 66
- PORT\_CNF\_OUT
  - dio.h, 66
- PORT\_A
  - dio.h, 66
- PORT\_B
  - dio.h, 66
- PORT\_C
  - dio.h, 66
- PORT\_D
  - dio.h, 67
- PORTA\_PTR
  - dio\_reg\_atm2560.h, 71
- PORTB\_CNF\_DDRB
  - dio\_port\_cnf.h, 68
- PORTB\_CNF\_PORTB
  - dio\_port\_cnf.h, 68

PORTB\_PTR  
     dio\_reg\_atm2560.h, 71  
 PORTC\_PTR  
     dio\_reg\_atm2560.h, 71  
 PORTD\_PTR  
     dio\_reg\_atm2560.h, 71  
 PRESCALER\_PERIODIC\_TIMER  
     scheduler.h, 85  
  
 read  
     dht22, 6  
 readTempSensor\_task  
     TempSensor, 24  
 removePeriodicTask  
     scheduler, 16  
  
 SW\_PERIOD\_MS  
     scheduler.h, 85  
 scheduler, 13  
     addPeriodicTask, 15  
     getPitNumber, 15  
     launchPeriodicTasks, 16  
     removePeriodicTask, 16  
     scheduler, 14  
     startScheduling, 17  
 scheduler.cpp  
     p\_scheduler, 83  
 scheduler.h  
     p\_scheduler, 86  
     PRESCALER\_PERIODIC\_TIMER, 85  
     SW\_PERIOD\_MS, 85  
     TIMER\_CTC\_VALUE, 86  
     TaskPtr\_t, 86  
 sendBool  
     UsartDebug, 36  
 sendData  
     UsartDebug, 37  
 sendInteger  
     UsartDebug, 38  
 setBaudRate  
     usart, 30  
 setValidity  
     TempSensor, 25  
 startScheduling  
     scheduler, 17  
 startTimer1  
     timer, 28  
 stopTimer1  
     timer, 28  
 str\_debug\_main\_menu  
     debug.cpp, 46  
  
 T\_ASW\_cnf\_struct, 18  
     p\_TempSensor, 19  
     p\_keepAliveLed, 19  
     p\_usartDebug, 19  
 T\_BSW\_cnf\_struct, 19  
     p\_dht22, 20  
     p\_dio, 20  
     p\_timer, 20  
     p\_usart, 20  
 TIMER\_CTC\_VALUE  
     scheduler.h, 86  
 TaskPtr\_t  
     scheduler.h, 86  
 TempSensor, 21  
     getHumPtr, 23  
     getHumidity, 22  
     getTemp, 23  
     getTempPtr, 24  
     readTempSensor\_task, 24  
     setValidity, 25  
     TempSensor, 21  
     updateLastValidValues, 26  
 TempSensor.cpp  
     PIT\_BEFORE\_INVALID, 52  
 TempSensor.h  
     PERIOD\_MS\_TASK\_TEMP\_SENSOR, 54  
 timer, 26  
     configureTimer1, 27  
     startTimer1, 28  
     stopTimer1, 28  
     timer, 27  
  
 USART\_BAUDRATE  
     bsw.h, 58  
 updateLastValidValues  
     TempSensor, 26  
 usart, 29  
     setBaudRate, 30  
     usart, 29  
     usart\_init, 30  
     usart\_read, 31  
     usart\_sendString, 31  
 usart\_init  
     usart, 30  
 usart\_read  
     usart, 31  
 usart\_sendString  
     usart, 31  
 UsartDebug, 32  
     activateDebugMode, 33  
     DebugModeManagement, 34  
     DisplaySensors\_task, 35  
     isDebugModeActive, 36  
     sendBool, 36  
     sendData, 37  
     sendInteger, 38  
     UsartDebug, 33  
  
 work/asw/TempSensor/TempSensor.cpp, 52  
 work/asw/TempSensor/TempSensor.h, 53  
 work/asw/asw.cpp, 41  
 work/asw/asw.h, 43  
 work/asw/debug/debug.cpp, 45  
 work/asw/debug/debug.h, 46  
 work/asw/keepAliveLed/keepAliveLed.cpp, 48  
 work/asw/keepAliveLed/keepAliveLed.h, 49

work/bsw/bsw.cpp, [55](#)  
work/bsw/bsw.h, [57](#)  
work/bsw/dht22/dht22.cpp, [60](#)  
work/bsw/dht22/dht22.h, [61](#)  
work/bsw/dio/dio.cpp, [63](#)  
work/bsw/dio/dio.h, [63](#)  
work/bsw/dio/dio\_port\_cnf.h, [67](#)  
work/bsw/dio/dio\_reg\_atm2560.h, [69](#)  
work/bsw/timer/timer.cpp, [72](#)  
work/bsw/timer/timer.h, [73](#)  
work/bsw/usart/usart.cpp, [73](#)  
work/bsw/usart/usart.h, [74](#)  
work/lib/operators.cpp, [75](#)  
work/lib/operators.h, [77](#)  
work/main.cpp, [78](#)  
work/main.h, [81](#)  
work/scheduler/scheduler.cpp, [83](#)  
work/scheduler/scheduler.h, [84](#)