

Arduino

1.0

Generated by Doxygen 1.8.14

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	dio Class Reference	5
3.1.1	Detailed Description	5
3.1.2	Constructor & Destructor Documentation	5
3.1.2.1	dio()	6
3.1.3	Member Function Documentation	6
3.1.3.1	dio_invertPortB()	6
3.1.3.2	dio_setPortB()	7
3.2	keepAliveLed Class Reference	7
3.2.1	Detailed Description	7
3.2.2	Constructor & Destructor Documentation	8
3.2.2.1	keepAliveLed()	8
3.2.3	Member Function Documentation	8
3.2.3.1	blinkLed_task()	8
3.3	scheduler Class Reference	9
3.3.1	Detailed Description	10
3.3.2	Constructor & Destructor Documentation	10
3.3.2.1	scheduler()	10

3.3.3	Member Function Documentation	10
3.3.3.1	addPeriodicTask()	10
3.3.3.2	launchPeriodicTasks()	11
3.3.3.3	startScheduling()	12
3.4	T_ASW_cnf_struct Struct Reference	13
3.4.1	Detailed Description	13
3.4.2	Member Data Documentation	13
3.4.2.1	p_keepAliveLed	13
3.4.2.2	p_usartDebug	14
3.5	T_BSW_cnf_struct Struct Reference	14
3.5.1	Detailed Description	14
3.5.2	Member Data Documentation	15
3.5.2.1	p_dio	15
3.5.2.2	p_timer	15
3.5.2.3	p_usart	15
3.6	timer Class Reference	15
3.6.1	Detailed Description	16
3.6.2	Constructor & Destructor Documentation	16
3.6.2.1	timer()	16
3.6.3	Member Function Documentation	16
3.6.3.1	configureTimer1()	16
3.6.3.2	startTimer1()	17
3.6.3.3	stopTimer1()	18
3.7	usart Class Reference	18
3.7.1	Detailed Description	18
3.7.2	Constructor & Destructor Documentation	18
3.7.2.1	usart()	18
3.7.3	Member Function Documentation	19
3.7.3.1	setBaudRate()	19
3.7.3.2	usart_init()	20
3.7.3.3	usart_sendString()	20
3.8	UsartDebug Class Reference	21
3.8.1	Detailed Description	21
3.8.2	Constructor & Destructor Documentation	21
3.8.2.1	UsartDebug()	21
3.8.3	Member Function Documentation	22
3.8.3.1	sendData()	22
3.8.3.2	sendInteger()	23

4	File Documentation	25
4.1	work/asw/asw.cpp File Reference	25
4.1.1	Detailed Description	26
4.1.2	Function Documentation	26
4.1.2.1	asw_init()	26
4.1.3	Variable Documentation	26
4.1.3.1	ASW_cnf_struct	27
4.2	work/asw/asw.h File Reference	27
4.2.1	Detailed Description	27
4.2.2	Function Documentation	28
4.2.2.1	asw_init()	28
4.2.3	Variable Documentation	28
4.2.3.1	ASW_cnf_struct	28
4.3	work/asw/keepAliveLed/keepAliveLed.cpp File Reference	29
4.3.1	Detailed Description	29
4.4	work/asw/keepAliveLed/keepAliveLed.h File Reference	29
4.4.1	Detailed Description	30
4.4.2	Macro Definition Documentation	30
4.4.2.1	PERIOD_MS_TASK_LED	30
4.5	work/asw/log/log.cpp File Reference	30
4.5.1	Detailed Description	31
4.6	work/asw/log/log.h File Reference	31
4.6.1	Detailed Description	31
4.7	work/bsw/bsw.cpp File Reference	32
4.7.1	Detailed Description	32
4.7.2	Function Documentation	32
4.7.2.1	bsw_init()	33
4.7.3	Variable Documentation	33
4.7.3.1	BSW_cnf_struct	33
4.8	work/bsw/bsw.h File Reference	33

4.8.1	Detailed Description	34
4.8.2	Macro Definition Documentation	34
4.8.2.1	USART_BAUDRATE	34
4.8.3	Function Documentation	34
4.8.3.1	bsw_init()	35
4.8.4	Variable Documentation	35
4.8.4.1	BSW_cnf_struct	35
4.9	work/bsw/dio/dio.cpp File Reference	35
4.9.1	Detailed Description	36
4.10	work/bsw/dio/dio.h File Reference	36
4.10.1	Detailed Description	36
4.11	work/bsw/timer/timer.cpp File Reference	37
4.11.1	Detailed Description	37
4.12	work/bsw/timer/timer.h File Reference	37
4.12.1	Detailed Description	38
4.13	work/bsw/usart/usart.cpp File Reference	38
4.13.1	Detailed Description	38
4.14	work/bsw/usart/usart.h File Reference	39
4.14.1	Detailed Description	39
4.15	work/lib/operators.cpp File Reference	39
4.15.1	Detailed Description	40
4.15.2	Function Documentation	40
4.15.2.1	operator delete()	40
4.15.2.2	operator new()	40
4.16	work/lib/operators.h File Reference	42
4.16.1	Detailed Description	42
4.16.2	Function Documentation	43
4.16.2.1	operator delete()	43
4.16.2.2	operator new()	43
4.17	work/main.cpp File Reference	43

4.17.1 Detailed Description	44
4.17.2 Function Documentation	44
4.17.2.1 ISR()	45
4.17.2.2 main()	45
4.18 work/main.h File Reference	46
4.18.1 Detailed Description	46
4.18.2 Macro Definition Documentation	46
4.18.2.1 DEBUG_FLAG	46
4.19 work/scheduler/scheduler.cpp File Reference	47
4.19.1 Detailed Description	47
4.19.2 Variable Documentation	47
4.19.2.1 p_scheduler	47
4.20 work/scheduler/scheduler.h File Reference	48
4.20.1 Detailed Description	48
4.20.2 Macro Definition Documentation	49
4.20.2.1 PRESCALER_PERIODIC_TIMER	49
4.20.2.2 SW_PERIOD_MS	49
4.20.2.3 TIMER_CTC_VALUE	49
4.20.3 Typedef Documentation	49
4.20.3.1 TaskPtr_t	49
4.20.4 Variable Documentation	49
4.20.4.1 p_scheduler	49
Index	51

Chapter 1

Class Index

1.1 Class List

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

dio	DIO class	5
keepAliveLed	Class for keep-alive LED blinking	7
scheduler	Scheduler class	9
T_ASW_cnf_struct	ASW configuration structure	13
T_BSW_cnf_struct	BSW configuration structure	14
timer	Class defining a timer	15
usart	USART serial bus class	18
UsartDebug	21

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

work/main.cpp		
Background task file	43	
work/main.h		
Background task header file	46	
work/asw/asw.cpp		
ASW main file	25	
work/asw/asw.h		
ASW main header file	27	
work/asw/keepAliveLed/keepAliveLed.cpp		
Definition of function for class keepAliveLed	29	
work/asw/keepAliveLed/keepAliveLed.h		
Class keepAliveLed header file	29	
work/asw/log/log.cpp		
This file defines classes for log and debug data transmission on USART link	30	
work/asw/log/log.h		
Header file for debug and logging functions	31	
work/bsw/bsw.cpp		
BSW main file	32	
work/bsw/bsw.h		
BSW main header file	33	
work/bsw/dio/dio.cpp		
DIO library	35	
work/bsw/dio/dio.h		
DIO library header file	36	
work/bsw/timer/timer.cpp		
Defines function for class timer	37	
work/bsw/timer/timer.h		
Timer class header file	37	
work/bsw/usart/usart.cpp		
BSW library for USART	38	
work/bsw/usart/usart.h		
Header file for USART library	39	
work/lib/operators.cpp		
C++ operators definitions	39	
work/lib/operators.h		
C++ operators definitions header file	42	

work/scheduler/ scheduler.cpp	
Defines scheduler class	47
work/scheduler/ scheduler.h	
Scheduler class header file	48

Chapter 3

Class Documentation

3.1 dio Class Reference

DIO class.

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

Public Member Functions

- [dio](#) ()
dio class constructor
- void [dio_setPortB](#) (uint8_t pin, bool state)
Port B setting function.
- void [dio_invertPortB](#) (uint8_t pin)
Inverts the state of output port.

3.1.1 Detailed Description

DIO class.

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

Definition at line 18 of file dio.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 dio()

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

dio class constructor

Initializes class dio and calls DIO hardware initialization function

Returns

Nothing

Definition at line 21 of file dio.cpp.

3.1.3 Member Function Documentation

3.1.3.1 dio_invertPortB()

```
void dio::dio_invertPortB (
    uint8_t pin )
```

Inverts the state of output port.

This function inverts the state of the chosen pin of port B

Parameters

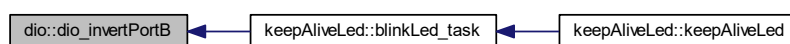
in	<i>pin</i>	Pin to invert
----	------------	---------------

Returns

Nothing

Definition at line 37 of file dio.cpp.

Here is the caller graph for this function:



3.1.3.2 dio_setPortB()

```
void dio::dio_setPortB (
    uint8_t pin,
    bool state )
```

Port B setting function.

This function sets the requested digital output on port B to the requested state

Parameters

in	<i>pin</i>	pin of PORT B to set
in	<i>state</i>	requested state to set pin

Returns

Nothing

Definition at line 26 of file dio.cpp.

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

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

3.2 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.

3.2.1 Detailed Description

Class for keep-alive LED blinking.

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

Definition at line 20 of file keepAliveLed.h.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 keepAliveLed()

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

Class constructor.

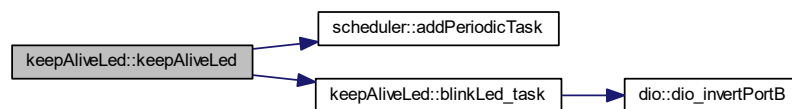
This function initializes the class keepAliveLed

Returns

Nothing

Definition at line 23 of file keepAliveLed.cpp.

Here is the call graph for this function:



3.2.3 Member Function Documentation

3.2.3.1 blinkLed_task()

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

Task for LED blinking.

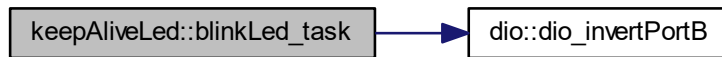
This function is inserted into the scheduler. It changes the state of the LED output to make it blink

Returns

Nothing

Definition at line 29 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](#)

3.3 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.

3.3.1 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 29 of file scheduler.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 scheduler()

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

scheduler class constructor

This function initializes the class scheduler

Returns

Nothing

Definition at line 28 of file scheduler.cpp.

Here is the call graph for this function:



3.3.3 Member Function Documentation

3.3.3.1 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

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 75 of file scheduler.cpp.

Here is the caller graph for this function:



3.3.3.2 launchPeriodicTasks()

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

Main scheduler function.

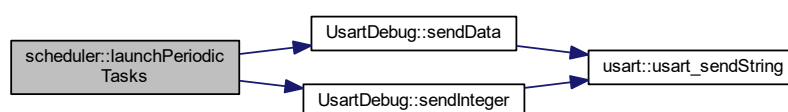
This function launches the scheduled tasks according to current software time and task configuration

Returns

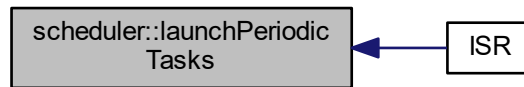
Nothing

Definition at line 41 of file scheduler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.3.3.3 startScheduling()

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

Starts the tasks scheduling.

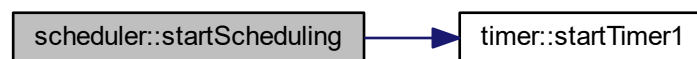
This function starts the timer which will trigger an interrupt every software period. When the interrupt is raised the scheduler will launch applications

Returns

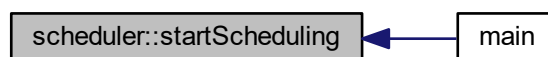
Nothing

Definition at line 69 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:

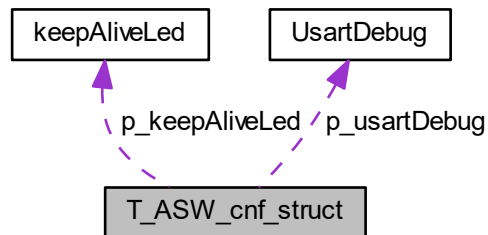
- [work/scheduler/scheduler.h](#)
- [work/scheduler/scheduler.cpp](#)

3.4 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`

3.4.1 Detailed Description

ASW configuration structure.

This structure contains all pointers to instanced applicative objects

Definition at line 19 of file `asw.h`.

3.4.2 Member Data Documentation

3.4.2.1 p_keepAliveLed

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

Pointer to `keepAliveLed` object

Definition at line 22 of file `asw.h`.

3.4.2.2 p_usartDebug

UsartDebug* T_ASW_cnf_struct::p_usartDebug

Pointer to usart debug object

Definition at line 21 of file asw.h.

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

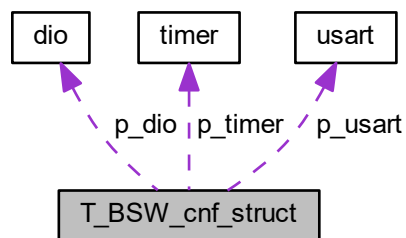
- work/asw/[asw.h](#)

3.5 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](#)

3.5.1 Detailed Description

BSW configuration structure.

This structure contains all pointers to instanced drivers objects

Definition at line 26 of file bsw.h.

3.5.2 Member Data Documentation

3.5.2.1 p_dio

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

Pointer to dio driver object

Definition at line 29 of file bsw.h.

3.5.2.2 p_timer

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

Pointer to timer driver object

Definition at line 30 of file bsw.h.

3.5.2.3 p_usart

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

Pointer to usart driver object

Definition at line 28 of file bsw.h.

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

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

3.6 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.

3.6.1 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.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 timer()

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

Class constructor.

This function initializes class attributes

Returns

Nothing

Definition at line 13 of file timer.cpp.

3.6.3 Member Function Documentation

3.6.3.1 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:



3.6.3.2 startTimer1()

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

Start Timer #1.

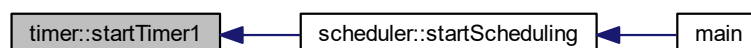
This functions starts Timer #1. Timer shall be initialized before this function is called.

Returns

Nothing

Definition at line 56 of file timer.cpp.

Here is the caller graph for this function:



3.6.3.3 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](#)

3.7 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.

3.7.1 Detailed Description

USART serial bus class.

This class defines all useful functions for USART serial bus

Definition at line 16 of file usart.h.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 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:



3.7.3 Member Function Documentation

3.7.3.1 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 62 of file usart.cpp.

3.7.3.2 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:



3.7.3.3 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

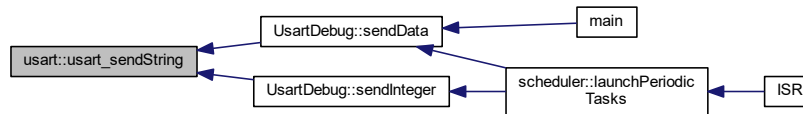
in	str	Pointer to the string being sent
----	-----	----------------------------------

Returns

Nothing.

Definition at line 43 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](#)

3.8 UsartDebug Class Reference

```
#include <log.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.

3.8.1 Detailed Description

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

Definition at line 18 of file log.h.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 UsartDebug()

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

Class [UsartDebug](#) constructor.

Initializes the class [UsartDebug](#)

Returns

Nothing

Definition at line 19 of file log.cpp.

3.8.3 Member Function Documentation

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

Returns

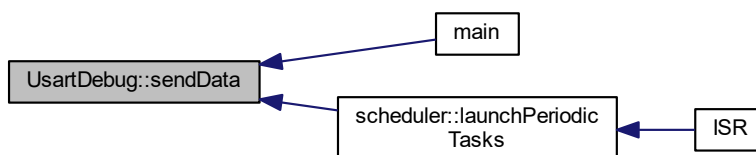
Nothing

Definition at line 24 of file log.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.8.3.2 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

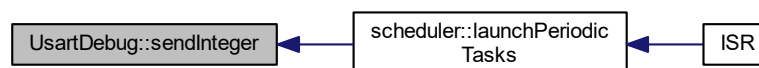
Nothing

Definition at line 30 of file log.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/log/log.h](#)
- [work/asw/log/log.cpp](#)

Chapter 4

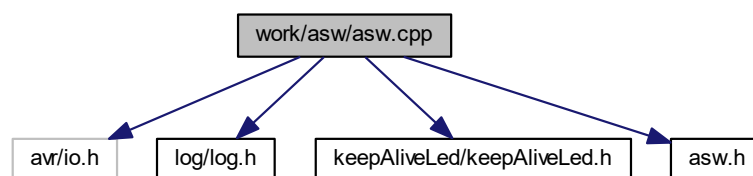
File Documentation

4.1 work/asw/asw.cpp File Reference

ASW main file.

```
#include <avr/io.h>
#include "log/log.h"
#include "keepAliveLed/keepAliveLed.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 22 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

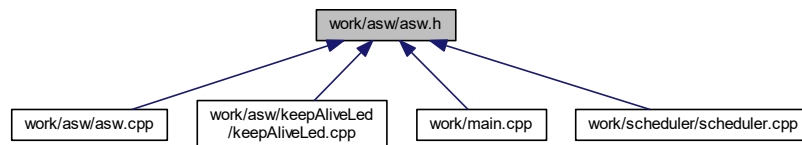
ASW configuration structure

Definition at line 19 of file asw.cpp.

4.2 work/asw/asw.h File Reference

ASW main header file.

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 22 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 19 of file asw.cpp.

4.3 work/asw/keepAliveLed/keepAliveLed.cpp File Reference

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

```
#include <avr/io.h>
#include "keepAliveLed.h"
#include "../../scheduler/scheduler.h"
#include "../../bsw/usart/usart.h"
#include "../../bsw/dio/dio.h"
#include "../../bsw/timer/timer.h"
#include "../../bsw/bsw.h"
#include "../log/log.h"
#include "../../asw/asw.h"
#include "../../main.h"
```

Include dependency graph for keepAliveLed.cpp:



4.3.1 Detailed Description

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

Date

17 mars 2018

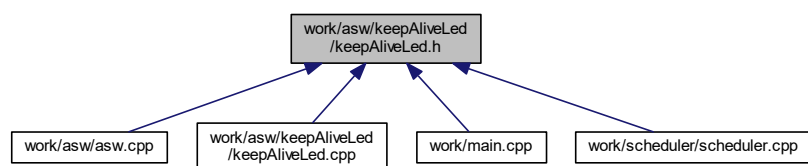
Author

nicls67

4.4 work/asw/keepAliveLed/keepAliveLed.h File Reference

Class [keepAliveLed](#) header file.

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`

4.4.1 Detailed Description

Class `keepAliveLed` header file.

Date

17 mars 2018

Author

nicls67

4.4.2 Macro Definition Documentation

4.4.2.1 PERIOD_MS_TASK_LED

```
#define PERIOD_MS_TASK_LED SW_PERIOD_MS
```

Period for led blinking

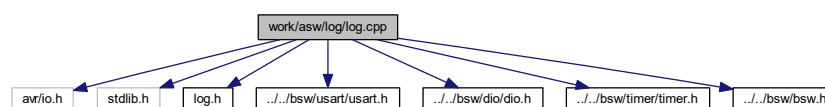
Definition at line 14 of file `keepAliveLed.h`.

4.5 work/asw/log/log.cpp File Reference

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

```
#include <avr/io.h>
#include <stdlib.h>
#include "log.h"
#include "../..bsw/usart/usart.h"
#include "../..bsw/dio/dio.h"
#include "../..bsw/timer/timer.h"
#include "../..bsw/bsw.h"
```

Include dependency graph for `log.cpp`:



4.5.1 Detailed Description

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

Date

15 mars 2018

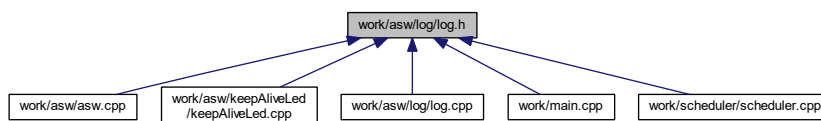
Author

nicls67

4.6 work/asw/log/log.h File Reference

Header file for debug and logging functions.

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



Classes

- class [UsartDebug](#)

4.6.1 Detailed Description

Header file for debug and logging functions.

Date

15 mars 2018

Author

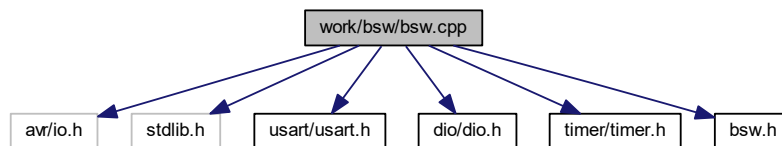
nicls67

4.7 work/bsw/bsw.cpp File Reference

BSW main file.

```
#include <avr/io.h>
#include <stdlib.h>
#include "usart/usart.h"
#include "dio/dio.h"
#include "timer/timer.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](#)

4.7.1 Detailed Description

BSW main file.

Date

13 mars 2018

Author

nicls67

4.7.2 Function Documentation

4.7.2.1 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 21 of file bsw.cpp.

Here is the caller graph for this function:



4.7.3 Variable Documentation

4.7.3.1 BSW_cnf_struct

```
T_BSW_cnf_struct BSW_cnf_struct
```

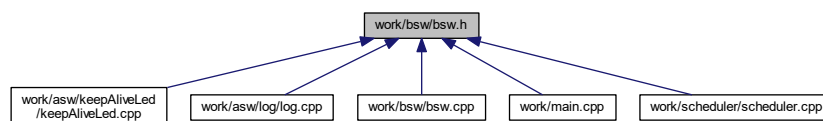
BSW configuration structure

Definition at line 19 of file bsw.cpp.

4.8 work/bsw/bsw.h File Reference

BSW main header file.

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](#)

4.8.1 Detailed Description

BSW main header file.

Date

13 mars 2018

Author

nicls67

4.8.2 Macro Definition Documentation

4.8.2.1 USART_BAUDRATE

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

usart connection to PC uses a baud rate of 9600

Definition at line 20 of file bsw.h.

4.8.3 Function Documentation

4.8.3.1 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 21 of file bsw.cpp.

Here is the caller graph for this function:



4.8.4 Variable Documentation

4.8.4.1 BSW_cnf_struct

`T_BSW_cnf_struct` BSW_cnf_struct

BSW configuration structure

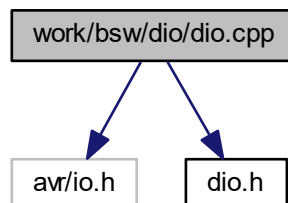
Definition at line 19 of file bsw.cpp.

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

DIO library.

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

Include dependency graph for dio.cpp:



4.9.1 Detailed Description

DIO library.

Date

13 mars 2018

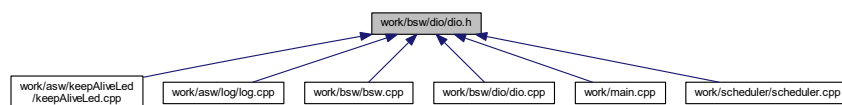
Author

nicls67

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

DIO library header file.

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



Classes

- class [dio](#)
DIO class.

4.10.1 Detailed Description

DIO library header file.

Date

13 mars 2018

Author

nicls67

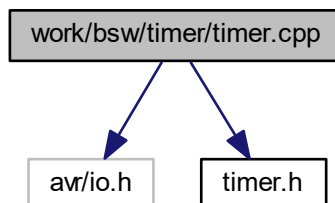
4.11 work/bsw/timer/timer.cpp File Reference

Defines function for class timer.

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

```
#include "timer.h"
```

Include dependency graph for timer.cpp:



4.11.1 Detailed Description

Defines function for class timer.

Date

15 mars 2018

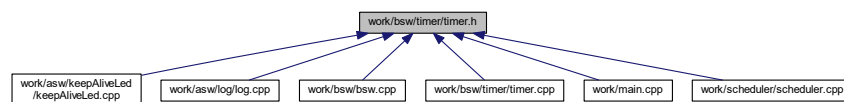
Author

nicls67

4.12 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.

4.12.1 Detailed Description

Timer class header file.

Date

15 mars 2018

Author

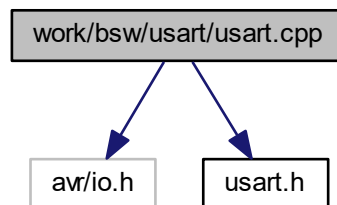
nicls67

4.13 work/bsw/usart/usart.cpp File Reference

BSW library for USART.

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

Include dependency graph for usart.cpp:



4.13.1 Detailed Description

BSW library for USART.

Date

13 mars 2018

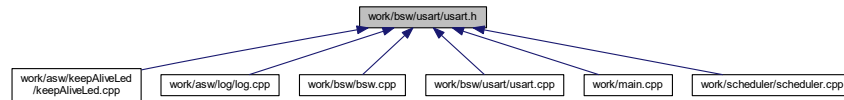
Author

nicls67

4.14 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.

4.14.1 Detailed Description

Header file for USART library.

Date

13 mars 2018

Author

nicls67

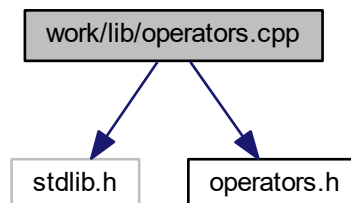
4.15 work/lib/operators.cpp File Reference

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.

4.15.1 Detailed Description

c++ operators definitions

Date

14 mars 2018

Author

nicls67

4.15.2 Function Documentation

4.15.2.1 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.

4.15.2.2 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	<code>a_size</code>	memory size to allocate
----	---------------------	-------------------------

Returns

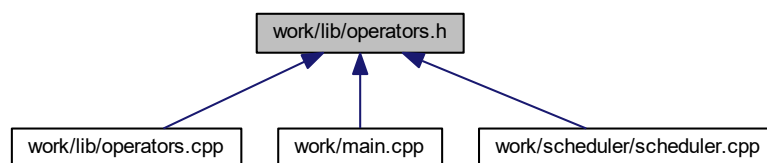
Pointer to the start of allocated memory zone

Definition at line 13 of file operators.cpp.

4.16 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.

4.16.1 Detailed Description

c++ operators definitions header file

Date

14 mars 2018

Author

nicls67

4.16.2 Function Documentation

4.16.2.1 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.

4.16.2.2 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.

4.17 work/main.cpp File Reference

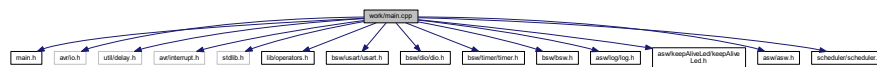
Background task file.

```

#include "main.h"
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include "lib/operators.h"
#include "bsw/usart/usart.h"
#include "bsw/dio/dio.h"
#include "bsw/timer/timer.h"
#include "bsw/bsw.h"
#include "asw/log/log.h"
#include "asw/keepAliveLed/keepAliveLed.h"
#include "asw/asw.h"
#include "scheduler/scheduler.h"

```

Include dependency graph for main.cpp:



Functions

- [ISR](#) (TIMER1_COMPA_vect)

Main software interrupt.

- int [main](#) (void)

Background task of program.

4.17.1 Detailed Description

Background task file.

Date

12 mars 2018

Author

nicls67

4.17.2 Function Documentation

4.17.2.1 ISR()

```
ISR (
    TIMER1_COMPA_vect )
```

Main software interrupt.

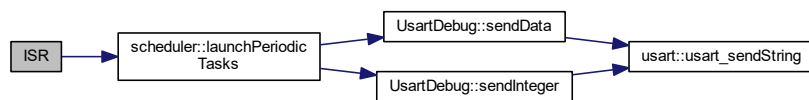
This function handles the interrupt raised by Timer #1. It wakes up the software every 500 ms to perform applications.

Returns

Nothing

Definition at line 35 of file main.cpp.

Here is the call graph for this function:



4.17.2.2 main()

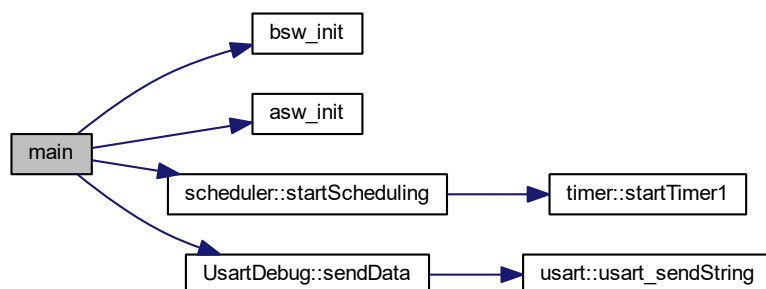
```
int main (
    void )
```

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 45 of file main.cpp.

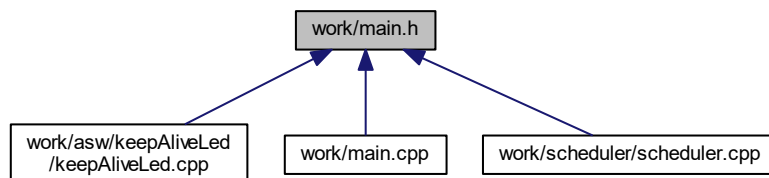
Here is the call graph for this function:



4.18 work/main.h File Reference

Background task header file.

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



Macros

- `#define` [DEBUG_FLAG](#)

4.18.1 Detailed Description

Background task header file.

Date

17 mars 2018

Author

nicls67

4.18.2 Macro Definition Documentation

4.18.2.1 DEBUG_FLAG

```
#define DEBUG_FLAG
```

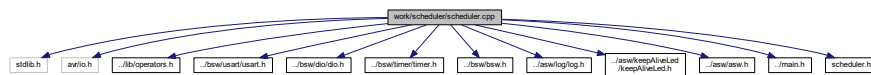
Definition at line 15 of file main.h.

4.19 work/scheduler/scheduler.cpp File Reference

Defines scheduler class.

```
#include <stdlib.h>
#include <avr/io.h>
#include "../lib/operators.h"
#include "../bsw/usart/usart.h"
#include "../bsw/dio/dio.h"
#include "../bsw/timer/timer.h"
#include "../bsw/bsw.h"
#include "../asw/log/log.h"
#include "../asw/keepAliveLed/keepAliveLed.h"
#include "../asw/asw.h"
#include "../main.h"
#include "scheduler.h"
```

Include dependency graph for scheduler.cpp:



Variables

- `scheduler * p_scheduler`

4.19.1 Detailed Description

Defines scheduler class.

Date

16 mars 2018

Author

nicls67

4.19.2 Variable Documentation

4.19.2.1 p_scheduler

`scheduler* p_scheduler`

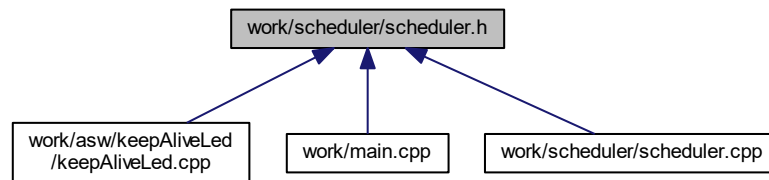
Pointer to scheduler object

Definition at line 26 of file scheduler.cpp.

4.20 work/scheduler/scheduler.h File Reference

Scheduler class header file.

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`

4.20.1 Detailed Description

Scheduler class header file.

Date

16 mars 2018

Author

nicls67

4.20.2 Macro Definition Documentation

4.20.2.1 PRESCALER_PERIODIC_TIMER

```
#define PRESCALER_PERIODIC_TIMER 256
```

Value of prescaler to use for periodic timer

Definition at line 16 of file scheduler.h.

4.20.2.2 SW_PERIOD_MS

```
#define SW_PERIOD_MS 500
```

Software period, used to define periodic timer interrupt

Definition at line 15 of file scheduler.h.

4.20.2.3 TIMER_CTC_VALUE

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

Compare value for periodic timer

Definition at line 17 of file scheduler.h.

4.20.3 Typedef Documentation

4.20.3.1 TaskPtr_t

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

Type defining a pointer to function.

Definition at line 22 of file scheduler.h.

4.20.4 Variable Documentation

4.20.4.1 p_scheduler

```
scheduler* p_scheduler
```

Pointer to scheduler object

Definition at line 26 of file scheduler.cpp.

Index

- ASW_cnf_struct
 - asw.cpp, 26
 - asw.h, 28
- addPeriodicTask
 - scheduler, 10
- asw.cpp
 - ASW_cnf_struct, 26
 - asw_init, 26
- asw.h
 - ASW_cnf_struct, 28
 - asw_init, 28
- asw_init
 - asw.cpp, 26
 - asw.h, 28
- BSW_cnf_struct
 - bsw.cpp, 33
 - bsw.h, 35
- blinkLed_task
 - keepAliveLed, 8
- bsw.cpp
 - BSW_cnf_struct, 33
 - bsw_init, 32
- bsw.h
 - BSW_cnf_struct, 35
 - bsw_init, 34
 - USART_BAUDRATE, 34
- bsw_init
 - bsw.cpp, 32
 - bsw.h, 34
- configureTimer1
 - timer, 16
- DEBUG_FLAG
 - main.h, 46
- dio, 5
 - dio, 5
 - dio_invertPortB, 6
 - dio_setPortB, 6
- dio_invertPortB
 - dio, 6
- dio_setPortB
 - dio, 6
- ISR
 - main.cpp, 44
- keepAliveLed, 7
 - blinkLed_task, 8
 - keepAliveLed, 8
- keepAliveLed.h
 - PERIOD_MS_TASK_LED, 30
- launchPeriodicTasks
 - scheduler, 11
- main
 - main.cpp, 45
- main.cpp
 - ISR, 44
 - main, 45
- main.h
 - DEBUG_FLAG, 46
- operator delete
 - operators.cpp, 40
 - operators.h, 43
- operator new
 - operators.cpp, 40
 - operators.h, 43
- operators.cpp
 - operator delete, 40
 - operator new, 40
- operators.h
 - operator delete, 43
 - operator new, 43
- p_dio
 - T_BSW_cnf_struct, 15
- p_keepAliveLed
 - T_ASW_cnf_struct, 13
- p_scheduler
 - scheduler.cpp, 47
 - scheduler.h, 49
- p_timer
 - T_BSW_cnf_struct, 15
- p_usart
 - T_BSW_cnf_struct, 15
- p_usartDebug
 - T_ASW_cnf_struct, 13
- PERIOD_MS_TASK_LED
 - keepAliveLed.h, 30
- PRESCALER_PERIODIC_TIMER
 - scheduler.h, 49
- SW_PERIOD_MS
 - scheduler.h, 49
- scheduler, 9
 - addPeriodicTask, 10
 - launchPeriodicTasks, 11
 - scheduler, 10

- startScheduling, 12
- scheduler.cpp
 - p_scheduler, 47
- scheduler.h
 - p_scheduler, 49
 - PRESCALER_PERIODIC_TIMER, 49
 - SW_PERIOD_MS, 49
 - TIMER_CTC_VALUE, 49
 - TaskPtr_t, 49
- sendData
 - UsartDebug, 22
- sendInteger
 - UsartDebug, 22
- setBaudRate
 - usart, 19
- startScheduling
 - scheduler, 12
- startTimer1
 - timer, 17
- stopTimer1
 - timer, 17
- T_ASW_cnf_struct, 13
 - p_keepAliveLed, 13
 - p_usartDebug, 13
- T_BSW_cnf_struct, 14
 - p_dio, 15
 - p_timer, 15
 - p_usart, 15
- TIMER_CTC_VALUE
 - scheduler.h, 49
- TaskPtr_t
 - scheduler.h, 49
- timer, 15
 - configureTimer1, 16
 - startTimer1, 17
 - stopTimer1, 17
 - timer, 16
- USART_BAUDRATE
 - bsw.h, 34
- usart, 18
 - setBaudRate, 19
 - usart, 18
 - usart_init, 19
 - usart_sendString, 20
- usart_init
 - usart, 19
- usart_sendString
 - usart, 20
- UsartDebug, 21
 - sendData, 22
 - sendInteger, 22
 - UsartDebug, 21
- work/asw/asw.cpp, 25
- work/asw/asw.h, 27
- work/asw/keepAliveLed/keepAliveLed.cpp, 29
- work/asw/keepAliveLed/keepAliveLed.h, 29
- work/asw/log/log.cpp, 30
- work/asw/log/log.h, 31
- work/bsw/bsw.cpp, 32
- work/bsw/bsw.h, 33
- work/bsw/dio/dio.cpp, 35
- work/bsw/dio/dio.h, 36
- work/bsw/timer/timer.cpp, 37
- work/bsw/timer/timer.h, 37
- work/bsw/usart/usart.cpp, 38
- work/bsw/usart/usart.h, 39
- work/lib/operators.cpp, 39
- work/lib/operators.h, 42
- work/main.cpp, 43
- work/main.h, 46
- work/scheduler/scheduler.cpp, 47
- work/scheduler/scheduler.h, 48