Arduino

1.0

Generated by Doxygen 1.8.14

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

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.

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 intialization function

Returns

Nothing

Definition at line 21 of file dio.cpp.

3.1.3 Member Function Documentation

3.1.3.1 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	*pin*	pin of PORT B to set
in	*state*	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:

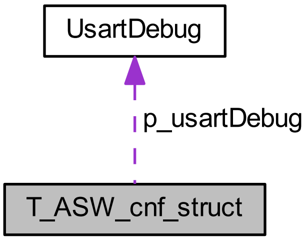
- work/bsw/dio/dio.h
- work/bsw/dio/dio.cpp

3.2 T_ASW_cnf_struct Struct Reference

ASW configuration structure.

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

Collaboration diagram for T_ASW_cnf_struct:



Public Attributes

- UsartDebug ∗ p_usartDebug

3.2.1 Detailed Description

ASW configuration structure.

This structure contains all pointers to instanced applicative objects

Definition at line 25 of file asw.h.

3.2.2 Member Data Documentation

3.2.2.1 p_usartDebug

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

Pointer to usart debug object

Definition at line 27 of file asw.h.

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

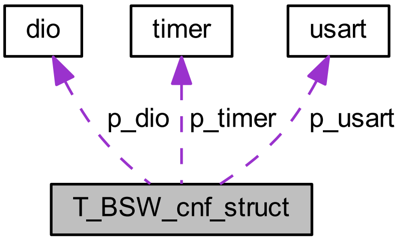
- work/asw/asw.h

3.3 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

3.3.1 Detailed Description

BSW configuration structure.

This structure contains all pointers to instanced drivers objects

Definition at line 30 of file bsw.h.

3.3.2 Member Data Documentation

3.3.2.1 p_dio

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

Pointer to dio driver object

Definition at line 33 of file bsw.h.

3.3.2.2 p_timer

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

Pointer to timer driver object

Definition at line 34 of file bsw.h.

3.3.2.3 p_usart

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

Pointer to usart driver object

Definition at line 32 of file bsw.h.

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

- work/bsw/bsw.h

3.4 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.

3.4.1 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.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 timer()

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

Class constructor.

This function initializes class attributes

Returns

Nothing

Definition at line 12 of file timer.cpp.

3.4.3 Member Function Documentation

3.4.3.1 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

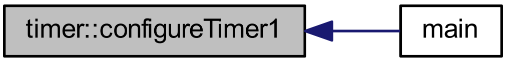
`in`	*a_prescaler*	prescaler value
`in`	*a_ctcValue*	Value to which the counter will compare before raising an interrupt

Returns

Nothing

Definition at line 17 of file timer.cpp.

Here is the caller graph for this function:



3.4.3.2 startTimer1()

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

Start Timer #1.

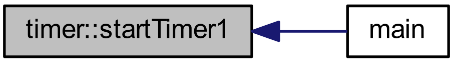
This functions starts Timer #1. Timer shall be initialized before this function is called.

Returns

Nothing

Definition at line 55 of file timer.cpp.

Here is the caller graph for this function:



3.4.3.3 stopTimer1()

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

Stops Timer #1.

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

Returns

Nothing

Definition at line 66 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

3.5 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.

3.5.1 Detailed Description

USART serial bus class.

This class defines all useful functions for USART serial bus

Definition at line 16 of file usart.h.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 usart()

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

Class usart constructor.

Initializes the class and call hardware initialization function

Parameters

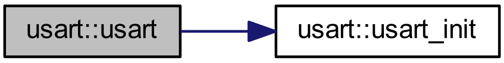
in	*a_BaudRate*	Desired Baud Rate (16 bit) - up to 57600

Returns

Nothing.

Definition at line 14 of file usart.cpp.

Here is the call graph for this function:



3.5.3 Member Function Documentation

3.5.3.1 setBaudRate()

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

Setting baud rate.

This function sets the attribute BaudRate of the class usart

Parameters

in	*a_BaudRate*	Desired Baud Rate (16 bit) - up to 57600

Returns

Nothing

Definition at line 62 of file usart.cpp.

3.5.3.2 usart_init()

```
void usart::usart_init ( )
```

USART hardware initialization.

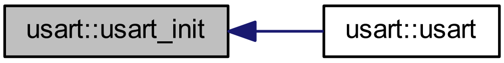
This function will initialize the USART using selected baudrate. User must pay attention to select one of the usually used Baud Rate (9600, 19200, 38400, 57600). Note that since an uint16 is used as argument, Baud rate cannot be more than 57600.

Returns

Nothing.

Definition at line 21 of file usart.cpp.

Here is the caller graph for this function:



3.5.3.3 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

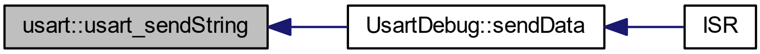
in	*str*	Pointer to the string being sent

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

3.6 UsartDebug Class Reference

```
#include <log.h>
```

Public Member Functions

- UsartDebug ()

  Class UsartDebug constructor.
- void sendData (uint8_t ∗str)

  Send a debug data on USART link.

3.6.1 Detailed Description

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

Definition at line 20 of file log.h.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 UsartDebug()

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

Class UsartDebug constructor.

Initializes the class UsartDebug

Returns

Nothing

Definition at line 12 of file log.cpp.

3.6.3 Member Function Documentation

3.6.3.1 sendData()

```
void UsartDebug::sendData (
             uint8_t * str )
```

Send a debug data on USART link.

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

Parameters

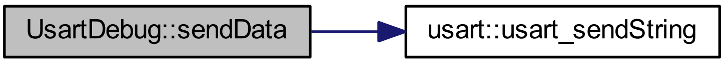
in	*str*	Pointer to the string being sent

Returns

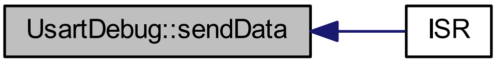
Nothing

Definition at line 17 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

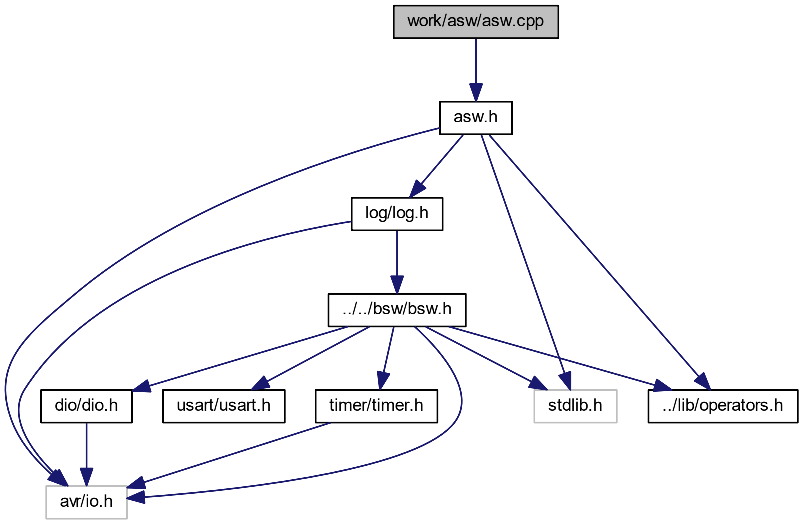
File Documentation

4.1 work/asw/asw.cpp File Reference

ASW main file.

```
#include "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.

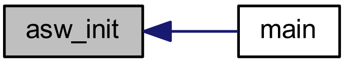
This function instantiates all applicative objects. The addresses of objects are then stored in ASW_cnf_struct structure. This function shall be called after BSW initialization function.

Returns

Nothing

Definition at line 16 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

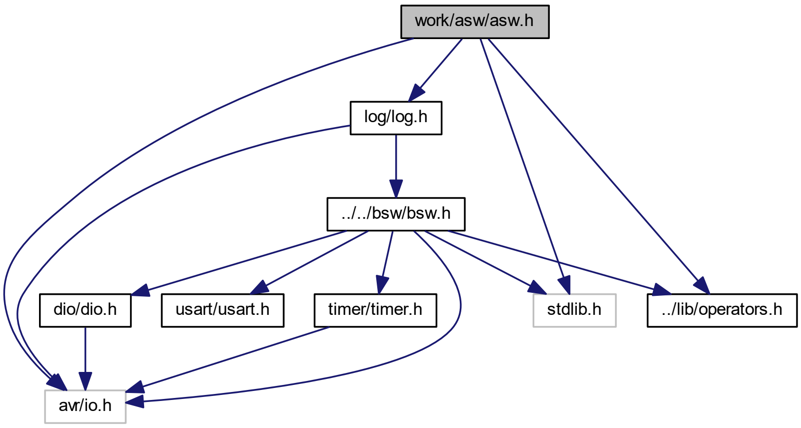
Definition at line 13 of file asw.cpp.

4.2 work/asw/asw.h File Reference

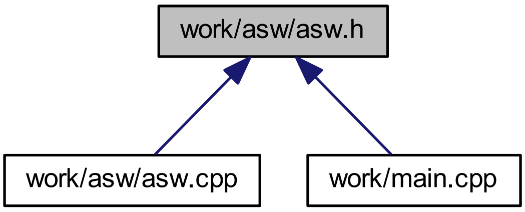
ASW main header file.

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

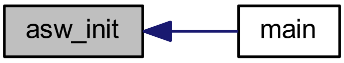
This function instantiates all applicative objects. The addresses of objects are then stored in ASW_cnf_struct structure. This function shall be called after BSW initialization function.

Returns

Nothing

Definition at line 16 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

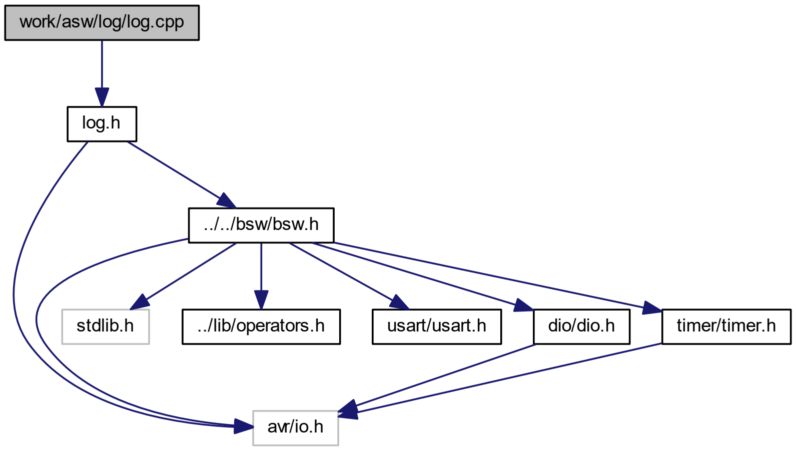
Definition at line 13 of file asw.cpp.

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

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

```
#include "log.h"
```

Include dependency graph for log.cpp:



4.3.1 Detailed Description

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

Date

15 mars 2018

Author

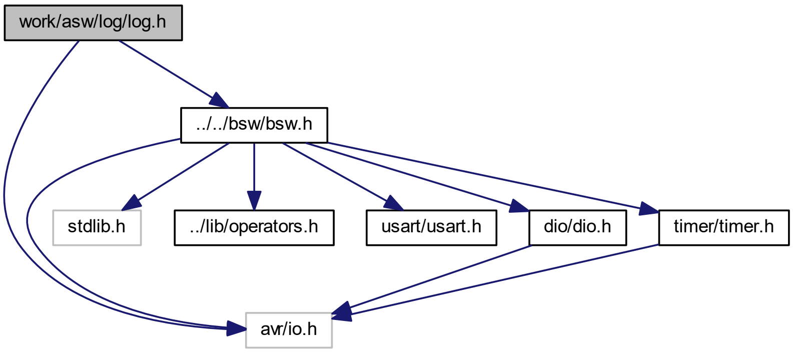
nicls67

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

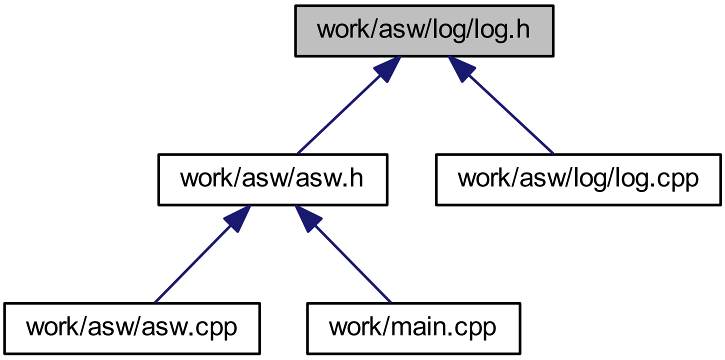
Header file for debug and logging functions.

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

Include dependency graph for log.h:



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



Classes

- class UsartDebug

4.4.1 Detailed Description

Header file for debug and logging functions.

Date

15 mars 2018

Author

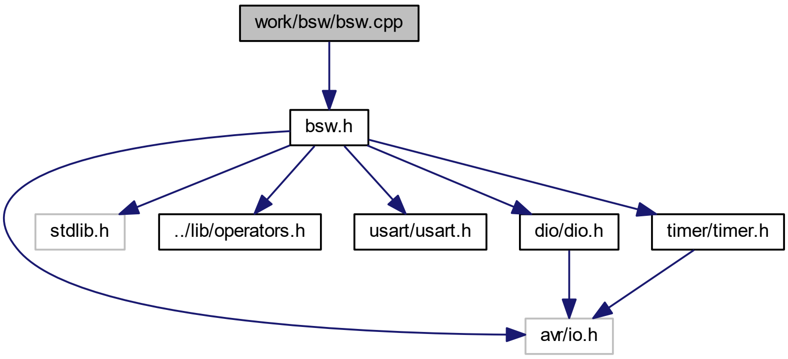
nicls67

4.5 work/bsw/bsw.cpp File Reference

BSW main file.

```
#include "bsw.h"
```

Include dependency graph for bsw.cpp:



Functions

- void bsw_init ()

    Initialization of BSW.

Variables

- T_BSW_cnf_struct BSW_cnf_struct

4.5.1 Detailed Description

BSW main file.

Date

13 mars 2018

Author

nicls67

4.5.2 Function Documentation

4.5.2.1 bsw_init()

```
void bsw_init ( )
```

Initialization of BSW.

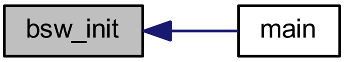
This function instantiates all driver objects, leading hardware initialization. The addresses of driver objects are then stored in BSW_cnf_struct structure.

Returns

Nothing

Definition at line 14 of file bsw.cpp.

Here is the caller graph for this function:



4.5.3 Variable Documentation

4.5.3.1 BSW_cnf_struct

```
T_BSW_cnf_struct BSW_cnf_struct
```

BSW configuration structure

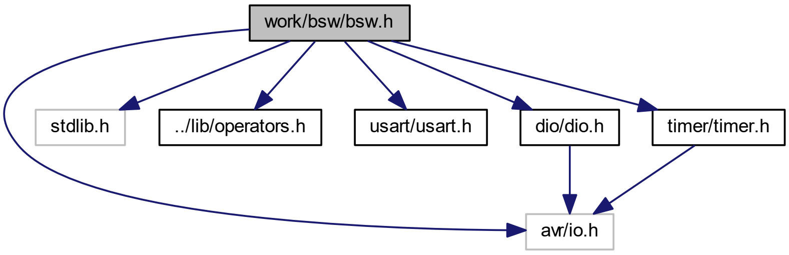
Definition at line 12 of file bsw.cpp.

4.6 work/bsw/bsw.h File Reference

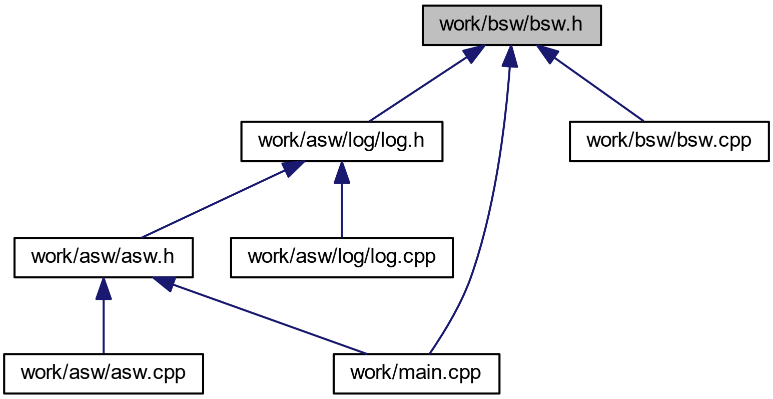
BSW main header file.

```
#include <avr/io.h>
#include <stdlib.h>
#include "../lib/operators.h"
#include "usart/usart.h"
#include "dio/dio.h"
#include "timer/timer.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

4.6.1 Detailed Description

BSW main header file.

Date

13 mars 2018

Author

nicls67

4.6.2 Macro Definition Documentation

4.6.2.1 USART_BAUDRATE

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

usart connection to PC uses a baud rate of 9600

Definition at line 24 of file bsw.h.

4.6.3 Function Documentation

4.6.3.1 bsw_init()

```
void bsw_init ( )
```

Initialization of BSW.

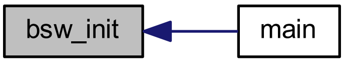
This function instantiates all driver objects, leading hardware initialization. The addresses of driver objects are then stored in BSW_cnf_struct structure.

Returns

Nothing

Definition at line 14 of file bsw.cpp.

Here is the caller graph for this function:



4.6.4 Variable Documentation

4.6.4.1 BSW_cnf_struct

```
T_BSW_cnf_struct BSW_cnf_struct
```

BSW configuration structure

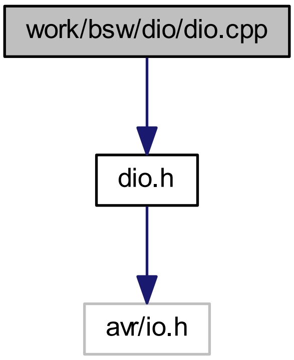
Definition at line 12 of file bsw.cpp.

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

DIO library.

```
#include "dio.h"
```

Include dependency graph for dio.cpp:



4.7.1 Detailed Description

DIO library.

Date

13 mars 2018

Author

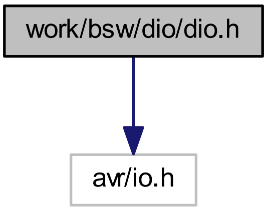
nicls67

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

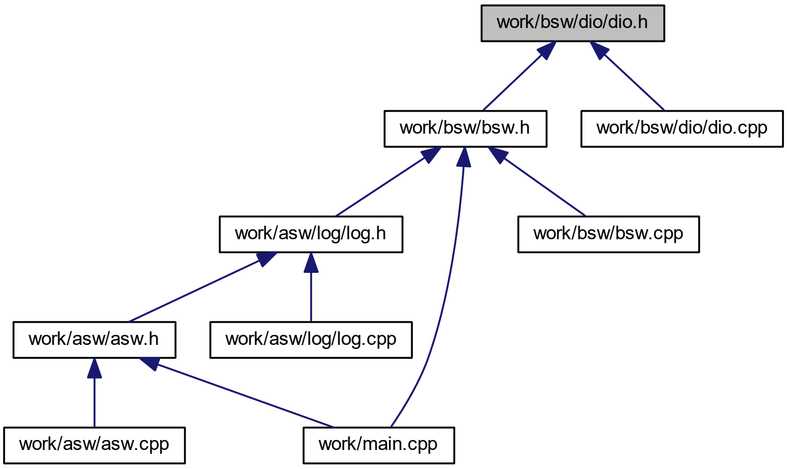
DIO library header file.

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

Include dependency graph for dio.h:



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



Classes

- class dio

  DIO class.

4.8.1 Detailed Description

DIO library header file.

Date

13 mars 2018

Author

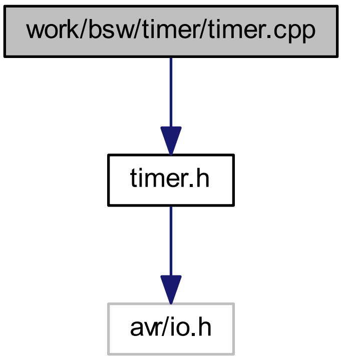
nicls67

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

Defines function for class timer.

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

Include dependency graph for timer.cpp:



4.9.1 Detailed Description

Defines function for class timer.

Date

15 mars 2018

Author

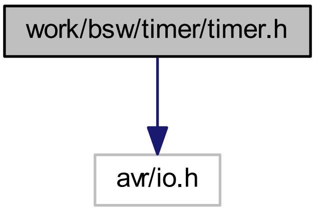
nicls67

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

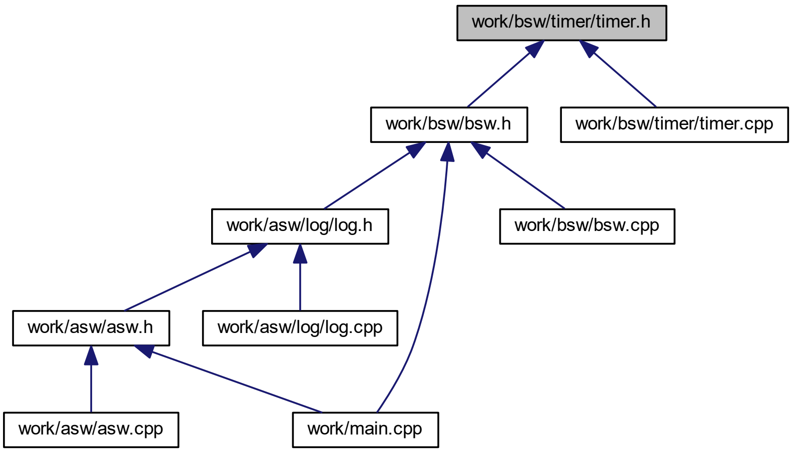
Timer class header file.

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

Include dependency graph for timer.h:



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



Classes

- class timer

  Class defining a timer.

4.10.1 Detailed Description

Timer class header file.

Date

15 mars 2018

Author

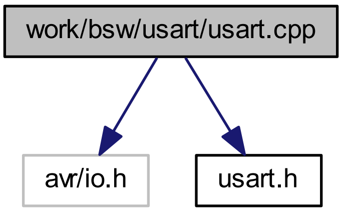
nicls67

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

BSW library for USART.

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

Include dependency graph for usart.cpp:



4.11.1 Detailed Description

BSW library for USART.

Date

13 mars 2018

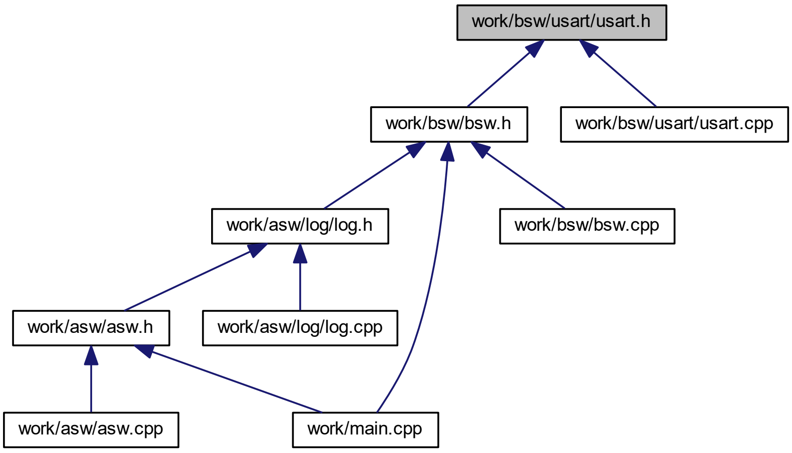
Author

nicls67

4.12 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.

4.12.1 Detailed Description

Header file for USART library.

Date

13 mars 2018

Author

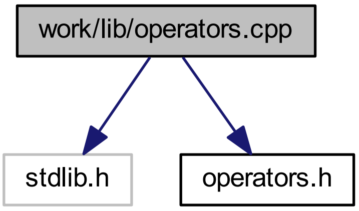
nicls67

4.13 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.

4.13.1 Detailed Description

c++ operators definitions

Date

14 mars 2018

Author

nicls67

4.13.2 Function Documentation

4.13.2.1 operator delete()

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

Operator delete.

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

Parameters

in	*ptr*	Pointer to the start of memory zone to free

Returns

Nothing

Definition at line 18 of file operators.cpp.

4.13.2.2 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	*a_size*	memory size to allocate

Returns

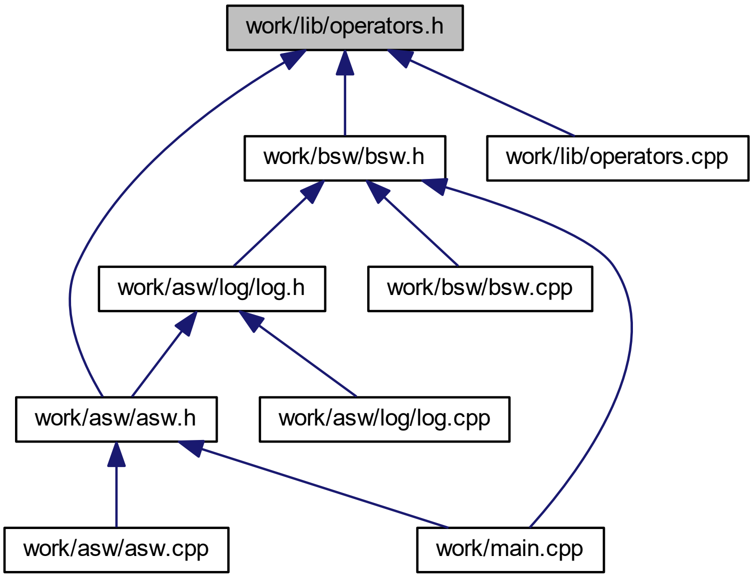
Pointer to the start of allocated memory zone

Definition at line 13 of file operators.cpp.

4.14 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.

4.14.1 Detailed Description

c++ operators definitions header file

Date

14 mars 2018

Author

nicls67

4.14.2 Function Documentation

4.14.2.1 operator delete()

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

Operator delete.

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

Parameters

in	*ptr*	Pointer to the start of memory zone to free

Returns

Nothing

Definition at line 18 of file operators.cpp.

4.14.2.2 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	*a_size*	memory size to allocate

Returns

Pointer to the start of allocated memory zone

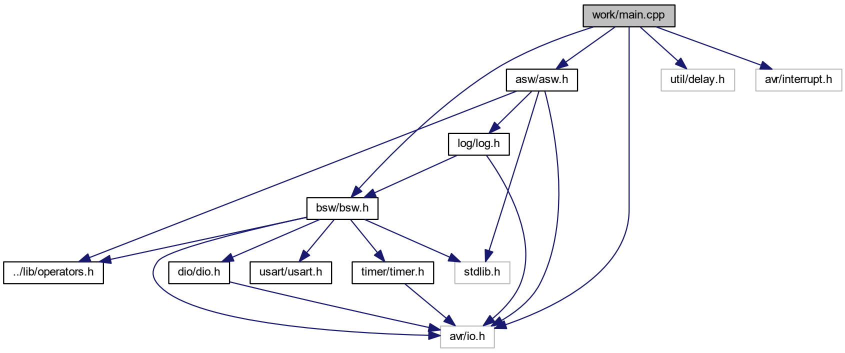
Definition at line 13 of file operators.cpp.

4.15 work/main.cpp File Reference

Background task file.

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

Include dependency graph for main.cpp:



Functions

- ISR (TIMER1_COMPA_vect)

  Main software interrupt.
- int main (void)

  Background task of program.

4.15.1 Detailed Description

Background task file.

Date

12 mars 2018

Author

nicls67

4.15.2 Function Documentation

4.15.2.1 ISR()

```
ISR (
            TIMER1_COMPA_vect   )
```

Main software interrupt.

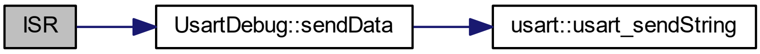
This function handles the interrupt raised by Timer #1. It wakes up the software every 500 ms to perform applications.

Returns

Nothing

Definition at line 24 of file main.cpp.

Here is the call graph for this function:



4.15.2.2 main()

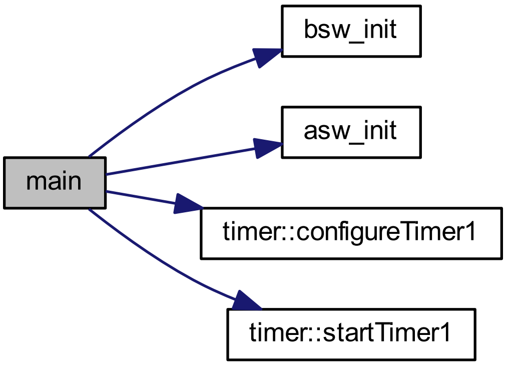
```
int main (
            void )
```

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 35 of file main.cpp.

Here is the call graph for this function:



Index