

usart_lib

v2

Generated by Doxygen 1.8.11

Contents

1	Hardware USART library for AVR 8bit MCU's	1
1.1	License	1
1.2	Introduction	1
1.2.1	Changelog	2
1.3	Usage	2
1.3.1	Interrupt based usage:	2
1.3.2	Normal usage:	3
2	README	5
3	Todo List	7
4	Bug List	9
5	Module Index	11
5.1	Modules	11
6	Data Structure Index	13
6.1	Data Structures	13
7	File Index	15
7.1	File List	15

8	Module Documentation	17
8.1	Macros	17
8.1.1	Detailed Description	18
8.1.2	Macro Definition Documentation	18
8.1.2.1	BAUD_CALC	18
8.1.2.2	USART0_RX_BUFFER_LENGTH	19
8.1.2.3	USART0_TX_BUFFER_LENGTH	19
8.1.2.4	USART1_RX_BUFFER_LENGTH	19
8.1.2.5	USART1_TX_BUFFER_LENGTH	19
8.1.2.6	USART2_RX_BUFFER_LENGTH	19
8.1.2.7	USART2_TX_BUFFER_LENGTH	20
8.1.2.8	USART3_RX_BUFFER_LENGTH	20
8.1.2.9	USART3_TX_BUFFER_LENGTH	20
8.2	Type definitions	21
8.2.1	Detailed Description	21
8.2.2	Enumeration Type Documentation	21
8.2.2.1	__txStatus	21
8.2.2.2	__usartNumber	21
8.3	Universal functions	22
8.3.1	Detailed Description	22
8.3.2	Function Documentation	22
8.3.2.1	usartInit(usartNumber_T const usartNumber, uint16_t const ubrrValue)	22
8.4	Interrupt mode receiver functions	23
8.4.1	Detailed Description	23
8.4.2	Function Documentation	23
8.4.2.1	registerRxBufferFullCallback(_usartFctPtr_T callback)	23
8.4.2.2	registerRxDataReadyCallback(_usartFctPtr_T callback)	23
8.4.2.3	usartGetByteFromReceiveBuffer(usartNumber_T const usartNumber)	23
8.4.2.4	usartRxStart(usartNumber_T const usartNumber)	24
8.5	Interrupt mode transmitter functions	25

8.5.1	Detailed Description	25
8.5.2	Function Documentation	25
8.5.2.1	registerTxCompleteCallback(_usartFctPtr_T callback)	25
8.5.2.2	usartPutByteToTransmitBuffer(usartNumber_T const usartNumber, uint8_t const data)	25
8.5.2.3	usartTxStart(usartNumber_T const usartNumber)	25
8.6	Normal mode receiver functions	27
8.6.1	Detailed Description	27
8.6.2	Function Documentation	27
8.6.2.1	usartDataReceived(usartNumber_T const usartNumber)	27
8.6.2.2	usartGetByte(usartNumber_T const usartNumber)	27
8.6.2.3	usartImGetByte(usartNumber_T const usartNumber)	27
8.7	Normal mode transmitter functions	29
8.7.1	Detailed Description	29
8.7.2	Function Documentation	29
8.7.2.1	usartDataTransferred(usartNumber_T const usartNumber)	29
8.7.2.2	usartImPutByte(usartNumber_T const usartNumber, uint8_t const data)	29
8.7.2.3	usartPutByte(usartNumber_T const usartNumber, uint8_t const data)	29
9	Data Structure Documentation	31
9.1	fifo_T Struct Reference	31
9.1.1	Detailed Description	31
9.1.2	Field Documentation	31
9.1.2.1	data	31
9.1.2.2	head	31
9.1.2.3	tail	32
9.2	usartTxBuffer_T Struct Reference	32
9.2.1	Detailed Description	32
9.2.2	Field Documentation	32
9.2.2.1	buffer	32
9.2.2.2	status	32
10	File Documentation	33
10.1	usart_lib-mach.h File Reference	33
10.1.1	Detailed Description	33
10.2	usart_lib.h File Reference	33
10.2.1	Detailed Description	36
11	Example Documentation	37
11.1	interrupt_mode.c	37
	Index	39

Chapter 1

Hardware USART library for AVR 8bit MCU's

Copyright

Piotr Rudzki (c)2015

Date

08.03.2016

1.1 License

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

For full license see <http://www.gnu.org/licenses/gpl-3.0.en.html>

1.2 Introduction

For now only supported data format is 8N1. This library supports two modes of operation for up to four USART's.

- Interrupt based mode with separate circular transmitting and receiving buffers for each enabled USART
- Normal mode without additional buffers.
- Mixing modes of operation is supported, e.g. USART0 in interrupt based mode, and USART1 in normal mode.

1.2.1 Changelog

Version

2.0 - 08.03.2016

- full library rewrite, whole usage change
- added doxygen generated documentation
- supported: ATmega162, ATmega48, ATmega88, ATmega168, ATmega328, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, Atmega16, ATmega32, ATmega323, ATmega8

Todo

- add support for more AVR MCU's
- add support for more frame formats
- add support for MPCM

Bug

- ATmega161 not working! Changes in [usartInit\(usartNumber_T, uint16_t\)](#) function and in [usart_lib-mach.h](#) needed.

1.3 Usage

- Include [usart_lib.h](#) in your sources.
- Define proper macros, below details.
- Desired USART must be initialized before use. In both modes procedure looks identical. Simply call [usartInit\(usartNumber_T, uint16_t\)](#) function.

1.3.1 Interrupt based usage:

- Define minimum one USART to use. e.g. `USE_USART0_INTERRUPT, USE_USART1_INTERRUPT, USE_USART2_INTERRUPT, USE_USART3_INTERRUPT`. You can simply define this at the beginning [usart_lib.h](#) or in CFLAGS passed to AVR-GCC. Second option is preferred.
- Additional define buffers length separate for every USART transmitter and receiver. e.g. `USART0_RX_BUFFER_LENGTH=32, USART0_TX_BUFFER_LENGTH=64`. NOTE buffer length must be power of 2 and not exceed 256. If you not define buffers for used USART both buffers will be 16 bytes length.
- Use functions provided for this purpose.

Warning

If interrupt mode isn't used for given USART it should not be enabled by macro `USE_USARTx_INTERRUPT`. It'll use some flash for two ISR (for receiver and transmitter) and some RAM for buffers!

See also

[Interrupt mode receiver functions](#)
[Interrupt mode transmitter functions](#)

1.3.2 Normal usage:

- Define minimum one USART to use: USE_USART0, USE_USART1, USE_USART2, USE_USART3. You can simply define this at the beginning [usart_lib.h](#) or in CFLAGS passed to AVR-GCC. Second option is preferred.
- Use functions provided for this purpose.

See also

[Normal mode receiver functions](#)

[Normal mode transmitter functions](#)

Chapter 2

README

License

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

For full license see <http://www.gnu.org/licenses/gpl-3.0.en.html> Introduction

For now only supported data format is 8N1. This library supports two modes of operation for up to four USART's.

Interrupt based mode with separate circular transmitting and receiving buffers for each enabled USART
Normal mode without additional buffers.
Mixing modes of operation is supported, e.g. USART0 in interrupt based mode, and USART1 in normal mode.

Changelog

Version 2.0 - 08.03.2016

full library rewrite, whole usage change
added doxygen generated documentation
supported: ATmega162, ATmega48, ATmega88, ATmega168, ATmega328, ATmega640, ATmega1280, ATmega1281, ATmega2560,

Todo:

add support for more AVR MCU's
add support for more frame formats
add support for MPCM

Bug:

ATmega161 not working! Changes in `usartInit(usartNumber_T, uint16_t)` function and in `usart_lib-mach.h` needed.

Usage

Include `usart_lib.h` in your sources.
Define proper macros, below details.
Desired USART must be initialized before use. In both modes procedure looks identical. Simply call `usartInit(u`

Interrupt based usage:

Define minimum one USART to use. e.g. `USE_USART0_INTERRUPT, USE_USART1_INTERRUPT, USE_USART1_INTERRUPT, USE_US`
Additional define buffers length separate for every USART transmitter and receiver. e.g. `USART0_RX_BUFFER_LEN`
Use functions provided for this purpose.

Warning If interrupt mode isn't used for given USART it should not be enabled by macro `USE_USARTx_INTERRUPT`. It'll use some flash for two ISR (for receiver and transmitter) and some RAM for buffers!

See also Interrupt mode receiver functions Interrupt mode transmitter functions

Normal usage:

Define minimum one USART to use: `USE_USART0, USE_USART1, USE_USART2, USE_USART3`. You can simply define this at
Use functions provided for this purpose.

See also Normal mode receiver functions Normal mode transmitter functions

Chapter 3

Todo List

page [Hardware USART library for AVR 8bit MCU's](#)

Chapter 4

Bug List

page [Hardware USART library for AVR 8bit MCU's](#)

Chapter 5

Module Index

5.1 Modules

Here is a list of all modules:

Macros	17
Type definitions	21
Universal functions	22
Interrupt mode receiver functions	23
Interrupt mode transmitter functions	25
Normal mode receiver functions	27
Normal mode transmitter functions	29

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

fifo_T	FIFO buffer type. Used only in interrupt based USART	31
usartTxBuffer_T	Transmitter structure. Used only in interrupt based USART	32

Chapter 7

File Index

7.1 File List

Here is a list of all documented files with brief descriptions:

usart_lib-mach.h	
Hardware support definitions	33
usart_lib.h	
Include usart_lib.h in Your sources. Do not edit this file, unless You know what You are doing .	33

Chapter 8

Module Documentation

8.1 Macros

Macros

- `#define USE_USART0`
define if You want USART0 support in normal mode
- `#define USE_USART1`
define if You want USART1 support in normal mode
- `#define USE_USART2`
define if You want USART2 support in normal mode
- `#define USE_USART3`
define if You want USART3 support in normal mode
- `#define USE_USART0_INTERRUPT`
define if You want USART0 support in interrupt mode
- `#define USE_USART1_INTERRUPT`
define if You want USART1 support in interrupt mode
- `#define USE_USART2_INTERRUPT`
define if You want USART2 support in interrupt mode
- `#define USE_USART3_INTERRUPT`
define if You want USART3 support in interrupt mode
- `#define USART0_RX_BUFFER_LENGTH 16`
USART0 receive buffer length used in interrupt mode.
- `#define USART0_TX_BUFFER_LENGTH 16`
USART0 transmitter buffer length used in interrupt mode.
- `#define USART1_RX_BUFFER_LENGTH 16`
USART1 receive buffer length used in interrupt mode.
- `#define USART1_TX_BUFFER_LENGTH 16`
USART1 transmitter buffer length used in interrupt mode.
- `#define USART2_RX_BUFFER_LENGTH 16`
USART2 receive buffer length used in interrupt mode.
- `#define USART2_TX_BUFFER_LENGTH 16`
USART2 transmitter buffer length used in interrupt mode.
- `#define USART3_RX_BUFFER_LENGTH 16`
USART3 receive buffer length used in interrupt mode.
- `#define USART3_TX_BUFFER_LENGTH 16`

- USART3 transmitter buffer length used in interrupt mode.*

• #define `ABS_VAL(x)` (((x) < 0LL) ? -(x) : (x))

Calculate absolute value for given signed long long. Used by `ERROR_CALC(x)`
- #define `UBRR_CALC(x)` (((F_CPU) + 8UL * (x)) / (16UL * (x)) - 1UL)

Calculate UBRR register value in normal mode. Used by `BAUD_CALC(x)`
- #define `DOUBLE_UBRR_CALC(x)` (((F_CPU) + 4UL * (x)) / (8UL * (x)) - 1UL)

Calculate UBRR register value in double mode. Used by `BAUD_CALC(x)`
- #define `CM_BAUD(x)` ((F_CPU) / (16UL * ((x) + 1UL)))

Calculate baud rate for given UBRR value in normal mode. Used by `BAUD_CALC(x)`
- #define `DOUBLE_CM_BAUD(x)` ((F_CPU) / (8UL * ((x) + 1UL)))

Calculate baud rate for given UBRR value in double mode. Used by `BAUD_CALC(x)`
- #define `ERROR_CALC(x, y)` (`ABS_VAL`((x) * 1000LL) / (y) - 1000LL)

Calculate baud rate error multiplied by 1000 for given close match baud rate x and desired y baud rate. Used by `BAUD_CALC(x)`
- #define `BAUD_CALC(x)`

Calculate UBRR register value for passed baud rate x.

8.1.1 Detailed Description

Macro definitions

8.1.2 Macro Definition Documentation

8.1.2.1 #define BAUD_CALC(x)

Value:

```
((ERROR_CALC(CM_BAUD(UBRR_CALC(x)), (x)) <= \
  ERROR_CALC(DOUBLE_CM_BAUD(DOUBLE_UBRR_CALC(x)), (x))) ? \
  UBRR_CALC(x) : \
  (DOUBLE_UBRR_CALC(x) | 0x8000))
```

Calculate UBRR register value for passed baud rate x.

If baud error in normal mode will be greater then baud error in double mode then macro returns UBRR value for double mode. Because maximum UBRR value must be lower then 4096 ($1 < 12$), macro sets 15th bit in returned value to indicate double mode.

Warning

This macro does not check for UBRR overflow!!! It doesn't test if baud rate error isn't too big!!! You should read datasheet for Your MCU to find out best baud rate for used F_CPU.

Examples:

`interrupt_mode.c.`

8.1.2.2 `#define USART0_RX_BUFFER_LENGTH 16`

USART0 receive buffer length used in interrupt mode.

Warning

maximum defined length 256

8.1.2.3 `#define USART0_TX_BUFFER_LENGTH 16`

USART0 transmitter buffer length used in interrupt mode.

Warning

maximum defined length 256

8.1.2.4 `#define USART1_RX_BUFFER_LENGTH 16`

USART1 receive buffer length used in interrupt mode.

Warning

maximum defined length 256

8.1.2.5 `#define USART1_TX_BUFFER_LENGTH 16`

USART1 transmitter buffer length used in interrupt mode.

Warning

maximum defined length 256

8.1.2.6 `#define USART2_RX_BUFFER_LENGTH 16`

USART2 receive buffer length used in interrupt mode.

Warning

maximum defined length 256

8.1.2.7 `#define USART2_TX_BUFFER_LENGTH 16`

USART2 transmitter buffer length used in interrupt mode.

Warning

maximum defined length 256

8.1.2.8 `#define USART3_RX_BUFFER_LENGTH 16`

USART3 receive buffer length used in interrupt mode.

Warning

maximum defined length 256

8.1.2.9 `#define USART3_TX_BUFFER_LENGTH 16`

USART3 transmitter buffer length used in interrupt mode.

Warning

maximum defined length 256

8.2 Type definitions

Data Structures

- struct [fifo_T](#)
FIFO buffer type. Used only in interrupt based USART.
- struct [usartTxBuffer_T](#)
Transmitter structure. Used only in interrupt based USART.

Typedefs

- typedef enum [__txStatus __txStatus_T](#)
USART transmitter status.
- typedef enum [__usartNumber usartNumber_T](#)
USART's names for use with library functions.
- typedef void(* [_usartFctPtr_T](#)) ([usartNumber_T](#) const)
Function pointer for library callbacks.

Enumerations

8.2.1 Detailed Description

New type definitions

8.2.2 Enumeration Type Documentation

8.2.2.1 enum [__txStatus](#)

USART transmitter status.

Enumerator

STOPPED library sets this when transmitter interrupt not working

STARTED library sets this when transmitter interrupt working

8.2.2.2 enum [__usartNumber](#)

USART's names for use with library functions.

Enumerator

USART0 for USART0

USART1 for USART1

USART2 for USART2

USART3 for USART3

8.3 Universal functions

Functions

- void `usartInit` (`usartNumber_T` const `usartNumber`, `uint16_t` const `ubrrValue`)
USART initialization.

8.3.1 Detailed Description

This group contains functions used by all modes transmitter and receiver

8.3.2 Function Documentation

8.3.2.1 void `usartInit` (`usartNumber_T` const *usartNumber*, `uint16_t` const *ubrrValue*)

USART initialization.

Always must be run for used USART. On the fly baud rate change supported. Simply use this function another time for desired USART. You should wait for all transmissions end before baud change.

Parameters

<i>usartNumber</i>	USART number (<code>usartNumber_T</code>)
<i>ubrrValue</i>	Value calculated with <code>BAUD_CALC(x)</code> macro

8.4 Interrupt mode receiver functions

Functions

- `int16_t usartGetByteFromReceiveBuffer (usartNumber_T const usartNumber)`
Get byte from receive buffer.
- `void registerRxDataReadyCallback (_usartFctPtr_T callback)`
Register callback function called when new data in buffer.
- `void registerRxBufferFullCallback (_usartFctPtr_T callback)`
Register callback function called when receive buffer full.
- `void usartRxStart (usartNumber_T const usartNumber)`
Start interrupt based receiver.

8.4.1 Detailed Description

Functions to use with interrupt mode receiver

8.4.2 Function Documentation

8.4.2.1 `void registerRxBufferFullCallback (_usartFctPtr_T callback)`

Register callback function called when receive buffer full.

Callback function must be void type, and get as argument USART number (`usartNumber_T`). Registering this function is not required.

Parameters

<i>callback</i>	Pointer to void function. Function must accept USART number as parameter (<code>usartNumber_T</code>)
-----------------	---

8.4.2.2 `void registerRxDataReadyCallback (_usartFctPtr_T callback)`

Register callback function called when new data in buffer.

Callback function must be void type, and get as argument USART number (`usartNumber_T`). Registering this function is not required.

Parameters

<i>callback</i>	Pointer to void function. Function must accept USART number as parameter (<code>usartNumber_T</code>)
-----------------	---

8.4.2.3 `int16_t usartGetByteFromReceiveBuffer (usartNumber_T const usartNumber)`

Get byte from receive buffer.

Returns

When buffer empty returns -1, otherwise returns data byte.

Parameters

<i>usartNumber</i>	USART number
--------------------	--------------

8.4.2.4 void usartRxStart (**usartNumber_T** const *usartNumber*)

Start interrupt based receiver.

Parameters

<i>usartNumber</i>	USART number (usartNumber_T)
--------------------	---------------------------------------

8.5 Interrupt mode transmitter functions

Functions

- `int8_t usartPutByteToTransmitBuffer (usartNumber_T const usartNumber, uint8_t const data)`
Put byte to transmit buffer.
- `void registerTxCompleteCallback (_usartFctPtr_T callback)`
Register callback function called when transmission from buffer ends.
- `void usartTxStart (usartNumber_T const usartNumber)`
Start interrupt based transmitter.

8.5.1 Detailed Description

Functions to use with interrupt mode transmitter

8.5.2 Function Documentation

8.5.2.1 void registerTxCompleteCallback (_usartFctPtr_T callback)

Register callback function called when transmission from buffer ends.

Callback function must be void type, and get as argument USART number (`usartNumber_T`). Registering this function is not required.

Parameters

<i>callback</i>	Pointer to void function. Function must accept USART number as parameter (<code>usartNumber_T</code>)
-----------------	---

8.5.2.2 int8_t usartPutByteToTransmitBuffer (usartNumber_T const usartNumber, uint8_t const data)

Put byte to transmit buffer.

Returns

When buffer full it doesn't put any data in and returns -1, otherwise returns 0.

Parameters

<i>usartNumber</i>	USART number
<i>data</i>	Byte to put in buffer

8.5.2.3 void usartTxStart (usartNumber_T const usartNumber)

Start interrupt based transmitter.

Parameters

<i>usartNumber</i>	USART number (usartNumber_T)
--------------------	------------------------------

8.6 Normal mode receiver functions

Functions

- `uint8_t usartDataReceived (usartNumber_T const usartNumber)`
Get receive complete flag.
- `uint8_t usartImGetByte (usartNumber_T const usartNumber)`
Immediate return contents of USART data register.
- `uint8_t usartGetByte (usartNumber_T const usartNumber)`
Wait for receive complete flag, then return contents of USART data register.

8.6.1 Detailed Description

Functions to use with normal mode receiver

8.6.2 Function Documentation

8.6.2.1 `uint8_t usartDataReceived (usartNumber_T const usartNumber)`

Get receive complete flag.

Returns

Returns non zero value if flag set, else returns 0

Parameters

<code>usartNumber</code>	USART number (usartNumber_T)
--------------------------	------------------------------

8.6.2.2 `uint8_t usartGetByte (usartNumber_T const usartNumber)`

Wait for receive complete flag, then return contents of USART data register.

Returns

USART data register contents

Parameters

<code>usartNumber</code>	USART number (usartNumber_T)
--------------------------	------------------------------

8.6.2.3 `uint8_t usartImGetByte (usartNumber_T const usartNumber)`

Immediate return contents of USART data register.

Returns

USART data register contents

Parameters

<i>usartNumber</i>	USART number (usartNumber_T)
--------------------	------------------------------

8.7 Normal mode transmitter functions

Functions

- `uint8_t usartDataTransferred (usartNumber_T const usartNumber)`
Get transmit complete flag.
- `void usartImPutByte (usartNumber_T const usartNumber, uint8_t const data)`
Immediate put byte to USART data register.
- `void usartPutByte (usartNumber_T const usartNumber, uint8_t const data)`
Wait for transmit complete flag, then put byte to USART data register.

8.7.1 Detailed Description

Functions to use with normal mode transmitter

8.7.2 Function Documentation

8.7.2.1 `uint8_t usartDataTransferred (usartNumber_T const usartNumber)`

Get transmit complete flag.

Returns

Returns non zero value if flag set, else returns 0

Parameters

<i>usartNumber</i>	USART number (usartNumber_T)
--------------------	------------------------------

8.7.2.2 `void usartImPutByte (usartNumber_T const usartNumber, uint8_t const data)`

Immediate put byte to USART data register.

Parameters

<i>usartNumber</i>	USART number (usartNumber_T)
<i>data</i>	Byte to put (uint8_t)

8.7.2.3 `void usartPutByte (usartNumber_T const usartNumber, uint8_t const data)`

Wait for transmit complete flag, then put byte to USART data register.

Parameters

<i>usartNumber</i>	USART number (usartNumber_T)
<i>data</i>	Byte to put (uint8_t)

Chapter 9

Data Structure Documentation

9.1 `fifo_T` Struct Reference

FIFO buffer type. Used only in interrupt based USART.

```
#include <usart_lib.h>
```

Data Fields

- volatile uint8_t `tail`
- volatile uint8_t `head`
- volatile uint8_t * `data`

9.1.1 Detailed Description

FIFO buffer type. Used only in interrupt based USART.

Maximum buffer capacity: 256 bytes.

9.1.2 Field Documentation

9.1.2.1 volatile uint8_t* `fifo_T::data`

pointer to buffer

9.1.2.2 volatile uint8_t `fifo_T::head`

last byte in buffer

9.1.2.3 volatile uint8_t fifo_T::tail

first byte in buffer

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

- [usart_lib.h](#)

9.2 usartTxBuffer_T Struct Reference

Transmitter structure. Used only in interrupt based USART.

```
#include <usart_lib.h>
```

Data Fields

- volatile [fifo_T](#) * [buffer](#)
- volatile [_txStatus_T](#) [status](#)

9.2.1 Detailed Description

Transmitter structure. Used only in interrupt based USART.

9.2.2 Field Documentation

9.2.2.1 volatile fifo_T* usartTxBuffer_T::buffer

pointer to buffer ([fifo_T](#))

9.2.2.2 volatile _txStatus_T usartTxBuffer_T::status

interrupt based transmitter status ([_txStarted_T](#))

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

- [usart_lib.h](#)

Chapter 10

File Documentation

10.1 `usart_lib-mach.h` File Reference

Hardware support definitions.

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

10.1.1 Detailed Description

Hardware support definitions.

If You want add support for new MCU you can edit this file. Read comments in code for details.

Copyright

Piotr Rudzki (c)2015

Date

08.03.2016

10.2 `usart_lib.h` File Reference

Include [usart_lib.h](#) in Your sources. Do not edit this file, unless You know what You are doing.

```
#include <avr/io.h>
#include "usart_lib-mach.h"
```

Data Structures

- struct [fifo_T](#)
FIFO buffer type. Used only in interrupt based USART.
- struct [usartTxBuffer_T](#)
Transmitter structure. Used only in interrupt based USART.

Macros

- `#define USE_USART0`
define if You want USART0 support in normal mode
- `#define USE_USART1`
define if You want USART1 support in normal mode
- `#define USE_USART2`
define if You want USART2 support in normal mode
- `#define USE_USART3`
define if You want USART3 support in normal mode
- `#define USE_USART0_INTERRUPT`
define if You want USART0 support in interrupt mode
- `#define USE_USART1_INTERRUPT`
define if You want USART1 support in interrupt mode
- `#define USE_USART2_INTERRUPT`
define if You want USART2 support in interrupt mode
- `#define USE_USART3_INTERRUPT`
define if You want USART3 support in interrupt mode
- `#define USART0_RX_BUFFER_LENGTH 16`
USART0 receive buffer length used in interrupt mode.
- `#define USART0_TX_BUFFER_LENGTH 16`
USART0 transmitter buffer length used in interrupt mode.
- `#define USART1_RX_BUFFER_LENGTH 16`
USART1 receive buffer length used in interrupt mode.
- `#define USART1_TX_BUFFER_LENGTH 16`
USART1 transmitter buffer length used in interrupt mode.
- `#define USART2_RX_BUFFER_LENGTH 16`
USART2 receive buffer length used in interrupt mode.
- `#define USART2_TX_BUFFER_LENGTH 16`
USART2 transmitter buffer length used in interrupt mode.
- `#define USART3_RX_BUFFER_LENGTH 16`
USART3 receive buffer length used in interrupt mode.
- `#define USART3_TX_BUFFER_LENGTH 16`
USART3 transmitter buffer length used in interrupt mode.
- `#define ABS_VAL(x) (((x) < 0LL) ? -(x) : (x))`
Calculate absolute value for given signed long long. Used by [ERROR_CALC\(x\)](#)
- `#define UBRR_CALC(x) (((F_CPU) + 8UL * (x)) / (16UL * (x)) - 1UL)`
Calculate UBRR register value in normal mode. Used by [BAUD_CALC\(x\)](#)
- `#define DOUBLE_UBRR_CALC(x) (((F_CPU) + 4UL * (x)) / (8UL * (x)) - 1UL)`
Calculate UBRR register value in double mode. Used by [BAUD_CALC\(x\)](#)
- `#define CM_BAUD(x) ((F_CPU) / (16UL * ((x) + 1UL)))`
Calculate baud rate for given UBRR value in normal mode. Used by [BAUD_CALC\(x\)](#)
- `#define DOUBLE_CM_BAUD(x) ((F_CPU) / (8UL * ((x) + 1UL)))`

- Calculate baud rate for given UBRR value in double mode. Used by [BAUD_CALC\(x\)](#)
- `#define ERROR_CALC(x, y) (ABS_VAL(((x) * 1000LL) / (y) - 1000LL))`

Calculate baud rate error multiplied by 1000 for given close match baud rate x and desired y baud rate. Used by [BAUD_CALC\(x\)](#)
- `#define BAUD_CALC(x)`

Calculate UBRR register value for passed baud rate x.

Typedefs

- `typedef enum __txStatus txStatus_T`

USART transmitter status.
- `typedef enum __usartNumber usartNumber_T`

USART's names for use with library functions.
- `typedef void(* _usartFctPtr_T) (usartNumber_T const)`

Function pointer for library callbacks.

Enumerations

Functions

- `int16_t usartGetByteFromReceiveBuffer (usartNumber_T const usartNumber)`

Get byte from receive buffer.
- `int8_t usartPutByteToTransmitBuffer (usartNumber_T const usartNumber, uint8_t const data)`

Put byte to transmit buffer.
- `void usartInit (usartNumber_T const usartNumber, uint16_t const ubrrValue)`

USART initialization.
- `void registerRxDataReadyCallback (_usartFctPtr_T callback)`

Register callback function called when new data in buffer.
- `void registerRxBufferFullCallback (_usartFctPtr_T callback)`

Register callback function called when receive buffer full.
- `void usartRxStart (usartNumber_T const usartNumber)`

Start interrupt based receiver.
- `uint8_t usartDataReceived (usartNumber_T const usartNumber)`

Get receive complete flag.
- `uint8_t usartImGetByte (usartNumber_T const usartNumber)`

Immediate return contents of USART data register.
- `uint8_t usartGetByte (usartNumber_T const usartNumber)`

Wait for receive complete flag, then return contents of USART data register.
- `void registerTxCompleteCallback (_usartFctPtr_T callback)`

Register callback function called when transmission from buffer ends.
- `void usartTxStart (usartNumber_T const usartNumber)`

Start interrupt based transmitter.
- `uint8_t usartDataTransferred (usartNumber_T const usartNumber)`

Get transmit complete flag.
- `void usartImPutByte (usartNumber_T const usartNumber, uint8_t const data)`

Immediate put byte to USART data register.
- `void usartPutByte (usartNumber_T const usartNumber, uint8_t const data)`

Wait for transmit complete flag, then put byte to USART data register.

10.2.1 Detailed Description

Include [usart_lib.h](#) in Your sources. Do not edit this file, unless You know what You are doing.

Copyright

Piotr Rudzki (c)2015

Date

08.03.2016

Chapter 11

Example Documentation

11.1 interrupt_mode.c

```
/*
 * interrupt_mode.c
 *
 * Created on: 08 mar 2016
 * Author: Piotr Rudzki ryba.lodz@gmail.com
 *
 * Simple interrupt mode example. It only echoes what it receives.
 * To test this example You must pass to compiler USE_USART0_INTERRUPT macro.
 * e.g. -DUSE_USART0_INTERRUPT
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 */

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

#include "usart_lib.h"

// usart_lib call this function when new data received
void rxDataReady(usartNumber_T const usartNumber) {
    int16_t tmp = usartGetByteFromReceiveBuffer(usartNumber); // get data from
    buffer
    if (tmp > -1) { // if buffer not empty
        int8_t txBufferFlag = usartPutByteToTransmitBuffer(usartNumber, (uint8_t
        )tmp); // put received data to transmit buffer
        if (txBufferFlag == 0) { // if there was room in buffer
            usartTxStart(usartNumber); // start transmitting data
        }
    }
}

int main(void) {
    usartInit(USART0, BAUD_CALC(14400)); // Initialize USART0
    registerRxDataReadyCallback(&rxDataReady); // Register callback
    usartRxStart(USART0); // Start interrupt based receiver

    sei(); // global interrupts enable

    // main program loop
    while (1) {
        // something to do without waiting for USART
    }
}
```


Index

- `__txStatus`
 - Type definitions, [21](#)
 - `__usartNumber`
 - Type definitions, [21](#)
- BAUD_CALC
 - Macros, [18](#)
- buffer
 - `usartTxBuffer_T`, [32](#)
- data
 - `fifo_T`, [31](#)
- `fifo_T`, [31](#)
 - data, [31](#)
 - head, [31](#)
 - tail, [31](#)
- head
 - `fifo_T`, [31](#)
- Interrupt mode receiver functions, [23](#)
 - `registerRxBufferFullCallback`, [23](#)
 - `registerRxDataReadyCallback`, [23](#)
 - `usartGetByteFromReceiveBuffer`, [23](#)
 - `usartRxStart`, [24](#)
- Interrupt mode transmitter functions, [25](#)
 - `registerTxCompleteCallback`, [25](#)
 - `usartPutByteToTransmitBuffer`, [25](#)
 - `usartTxStart`, [25](#)
- Macros, [17](#)
 - BAUD_CALC, [18](#)
 - USART0_RX_BUFFER_LENGTH, [18](#)
 - USART0_TX_BUFFER_LENGTH, [19](#)
 - USART1_RX_BUFFER_LENGTH, [19](#)
 - USART1_TX_BUFFER_LENGTH, [19](#)
 - USART2_RX_BUFFER_LENGTH, [19](#)
 - USART2_TX_BUFFER_LENGTH, [19](#)
 - USART3_RX_BUFFER_LENGTH, [20](#)
 - USART3_TX_BUFFER_LENGTH, [20](#)
- Normal mode receiver functions, [27](#)
 - `usartDataReceived`, [27](#)
 - `usartGetByte`, [27](#)
 - `usartImGetByte`, [27](#)
- Normal mode transmitter functions, [29](#)
 - `usartDataTransferred`, [29](#)
 - `usartImPutByte`, [29](#)
 - `usartPutByte`, [29](#)
- `registerRxBufferFullCallback`
 - Interrupt mode receiver functions, [23](#)
- `registerRxDataReadyCallback`
 - Interrupt mode receiver functions, [23](#)
- `registerTxCompleteCallback`
 - Interrupt mode transmitter functions, [25](#)
- STARTED
 - Type definitions, [21](#)
- STOPPED
 - Type definitions, [21](#)
- status
 - `usartTxBuffer_T`, [32](#)
- tail
 - `fifo_T`, [31](#)
- Type definitions, [21](#)
 - `__txStatus`, [21](#)
 - `__usartNumber`, [21](#)
 - STARTED, [21](#)
 - STOPPED, [21](#)
 - USART0, [21](#)
 - USART1, [21](#)
 - USART2, [21](#)
 - USART3, [21](#)
- USART0
 - Type definitions, [21](#)
- USART0_RX_BUFFER_LENGTH
 - Macros, [18](#)
- USART0_TX_BUFFER_LENGTH
 - Macros, [19](#)
- USART1
 - Type definitions, [21](#)
- USART1_RX_BUFFER_LENGTH
 - Macros, [19](#)
- USART1_TX_BUFFER_LENGTH
 - Macros, [19](#)
- USART2
 - Type definitions, [21](#)
- USART2_RX_BUFFER_LENGTH
 - Macros, [19](#)
- USART2_TX_BUFFER_LENGTH
 - Macros, [19](#)
- USART3
 - Type definitions, [21](#)
- USART3_RX_BUFFER_LENGTH
 - Macros, [20](#)
- USART3_TX_BUFFER_LENGTH
 - Macros, [20](#)

- Universal functions, [22](#)
 - usartInit, [22](#)
- usart_lib-mach.h, [33](#)
- usart_lib.h, [33](#)
- usartDataReceived
 - Normal mode receiver functions, [27](#)
- usartDataTransferred
 - Normal mode transmitter functions, [29](#)
- usartGetByte
 - Normal mode receiver functions, [27](#)
- usartGetByteFromReceiveBuffer
 - Interrupt mode receiver functions, [23](#)
- usartImGetByte
 - Normal mode receiver functions, [27](#)
- usartImPutByte
 - Normal mode transmitter functions, [29](#)
- usartInit
 - Universal functions, [22](#)
- usartPutByte
 - Normal mode transmitter functions, [29](#)
- usartPutByteToTransmitBuffer
 - Interrupt mode transmitter functions, [25](#)
- usartRxStart
 - Interrupt mode receiver functions, [24](#)
- usartTxBuffer_T, [32](#)
 - buffer, [32](#)
 - status, [32](#)
- usartTxStart
 - Interrupt mode transmitter functions, [25](#)