

UartDisplay

1.0

Generated by Doxygen 1.7.2

Tue Feb 15 2011 18:32:45

Contents

1	UartDisplay	1
2	ATMega USART Stack	3
2.1	Introduction	3
2.2	Usage	3
2.2.1	Using the USART as stdout	3
3	Todo List	5
4	Bug List	7
5	File Index	9
5.1	File List	9
6	File Documentation	11
6.1	lcd.c File Reference	11
6.1.1	Detailed Description	13
6.1.2	Function Documentation	14
6.1.2.1	lcd_putchar	14
6.1.2.2	lcd_scroll	14
6.1.2.3	print_buffer	15
6.1.2.4	push_char	16
6.1.3	Variable Documentation	17
6.1.3.1	input_buffer	17
6.1.3.2	remap	17
6.2	lcd.h File Reference	18
6.2.1	Detailed Description	21
6.2.2	Define Documentation	21
6.2.2.1	LCD_TYPE	21
6.2.3	Function Documentation	22
6.2.3.1	lcd_command	22
6.2.3.2	lcd_data	22
6.2.3.3	lcd_init	22
6.2.3.4	lcd_putchar	22
6.2.3.5	lcd_puts	23
6.2.3.6	lcd_puts_P	24
6.2.3.7	lcd_readadr	24
6.2.3.8	lcd_scroll	24
6.2.3.9	lcd_setadr	25
6.2.3.10	print_buffer	25

6.3	uart.c File Reference	26
6.3.1	Detailed Description	28
6.3.2	Function Documentation	29
6.3.2.1	ISR	29
6.3.2.2	ISR	29
6.3.2.3	uart_getbyte	29
6.3.2.4	uart_getchar	30
6.3.2.5	uart_init	30
6.3.2.6	uart_putchar	31
6.3.2.7	uart_sendbyte	31
6.3.2.8	uart_senddata	32
6.3.2.9	uart_sendid	32
6.4	uart.h File Reference	33
6.4.1	Detailed Description	35
6.4.2	Define Documentation	36
6.4.2.1	UART_CHRSIZE_9BIT	36
6.4.2.2	UART_MPCM_ENABLE	36
6.4.2.3	UART_PRINTF_COMPATIBILITY	36
6.4.2.4	UART_RX_BUFFER_SIZE	36
6.4.2.5	UART_TX_BUFFER_SIZE	36
6.4.2.6	UBRR_val	36
6.4.3	Function Documentation	37
6.4.3.1	uart_getbyte	37
6.4.3.2	uart_getchar	37
6.4.3.3	uart_init	38
6.4.3.4	uart_putchar	38
6.4.3.5	uart_sendbyte	39
6.4.3.6	uart_senddata	40
6.4.3.7	uart_sendid	40
6.5	UartDisplay.c File Reference	40
6.5.1	Detailed Description	41
6.5.2	Function Documentation	42
6.5.2.1	timer2_init	42

Chapter 1

UartDisplay

This Project implements the Dotmatrix LCD Library written in Assembler and provides a UART Interface to a PC or other Mikrocontrollers.

The Usage is straight forward, just send a string, terminated by

. This string is then displayed on the last line of the screen. The previous content of the screen is shifted up by one line

See <https://wiki.fs-et.de/student-lab/UartDisplay> for more details (German).

Chapter 2

ATMega USART Stack

2.1 Introduction

This File implements a set of functions to send and receive data through the hardware USART Module of a AVR ATMega Device.

Data transfer is fully interrupt driven and fifo buffered.

2.2 Usage

To use this USART Stack just copy [uart.c](#) and [uart.h](#) to your project directory, include [uart.h](#) wherever you want and adjust the configuration to your needs.

2.2.1 Using the USART as stdout

If you want to use stdio printf functions to write formatted text to the UART you may want to include the following code:

```
#include <stdio.h>
#include <avr/pgmspace.h>
#include "uart0.h"

static FILE pc_out = FDEV_SETUP_STREAM(uart0_putchar, NULL, _FDEV_SETUP_WRITE);
#define pc_print(fmt,...) fprintf_P(&pc_out, PSTR(fmt), ##__VA_ARGS__)
```


Chapter 3

Todo List

- File [UartDisplay.c](#)
- Implement Buttons
 - implement Buzzer

Chapter 4

Bug List

File [lcd.c](#) • "o" is not displayed correctly

Chapter 5

File Index

5.1 File List

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

lcd.c (Dotmatrix LCD Library)	11
lcd.h (Dotmatrix LCD Library)	18
uart.c (Implementation of USART Stack)	26
uart.h (Implementation of USART Stack)	33
UartDisplay.c (UartDisplay Main File)	40

Chapter 6

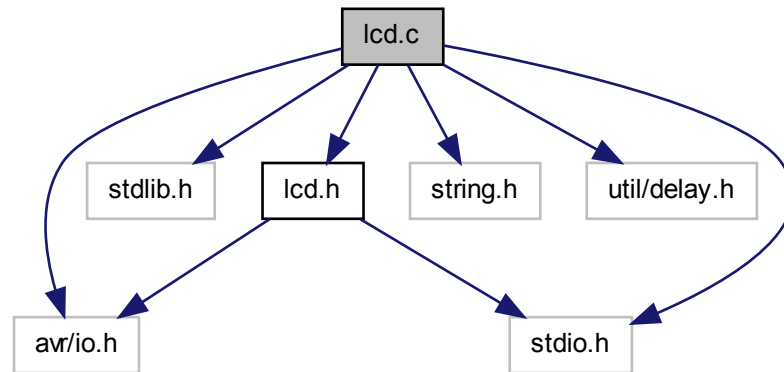
File Documentation

6.1 lcd.c File Reference

Dotmatrix LCD Library.

```
#include <avr/io.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <util/delay.h>
#include "lcd.h"
```

Include dependency graph for lcd.c:



Functions

- void `print_buffer` (void)
send the buffer contents to the display
- void `lcd_scroll` (void)
scroll individual lines
- void `push_char` (uint8_t data)
push a character into the buffer.
- int `lcd_putchar` (char data, FILE *stream)
Write one Character on the Display.

Variables

- char `display_buffer` [LCD_ROWS][LCD_COLS+1]
buffer used when scrolling is enabled
- char `input_buffer` [LCD_ROWS][INPUT_BUFFER_SIZE+1]
buffer used to store the received characters and rotate the lines
- uint8_t `buffer_index` = 0

always points to the first line

- uint8_t `char_index` = 0

points to the next char to write

- uint8_t `formfeed` = 0

is set if a formfeed () is received to parse a sequence of special characters

- uint8_t `escape` = 0

is set if an escape is received, to parse control sequences

- uint8_t `firstchar` = 0

contains the first char of a sequence of 2 chars that should be parsed

- uint8_t `scroll` = 0

scroll enable flag

- char `remap` [256]

lookup table to translate display charset to ASCII charset

6.1.1 Detailed Description

Dotmatrix LCD Library.

Authors

Dominic Rathje (dominic.rathje@uni-ulm.de)

Christian Degenhart (christian.degenhart@uni-ulm.de)

Version

1.0

Note

- Compiler : WinAVR 20100110
- Supported devices : ATmega8

Bug

- "°" is not displayed correctly

Definition in file [lcd.c](#).

6.1.2 Function Documentation

6.1.2.1 int lcd_putchar (char *data*, FILE * *stream*)

Write one Character on the Display.

Characters received are transformer by a lookup-table because the charset of the Display is not equal to ASCII. As input to this function any ASCII character that can be typed ba a German or American Keyboard is valid and displayed correctly on the screen.

As the Display has some amazing special characters, like fancy arrows, there are some sequences that are parsed and replaced by the corresponding symbol. See <https://wiki.fs-et.de/student-lab/UartDisplay> for details on the parser.

Parameters

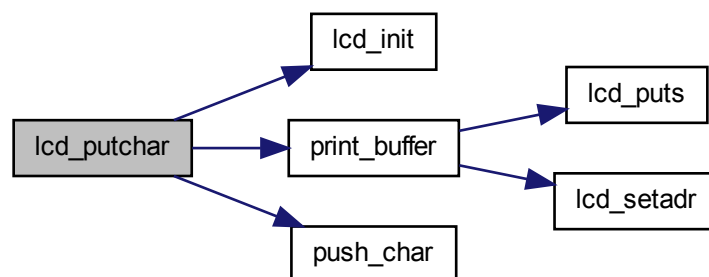
<i>data</i>	ASCII Character to print.
<i>stream</i>	(is necessary to be compatible with printf)

Returns

0 on success.

Definition at line 181 of file lcd.c.

Here is the call graph for this function:



6.1.2.2 void lcd_scroll (void)

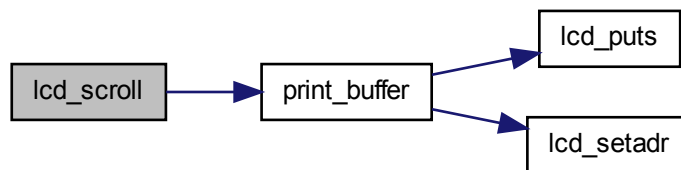
scroll individual lines

This function should be called by the main loop regularly if scroll mode is activated.

Each time this function is executed the content of a line longer than the physical display length is rotated by one position.

Definition at line 128 of file lcd.c.

Here is the call graph for this function:



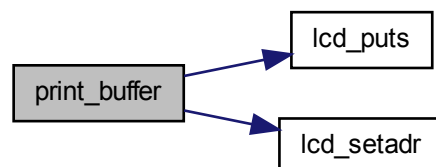
6.1.2.3 void print_buffer (void)

send the buffer contents to the display

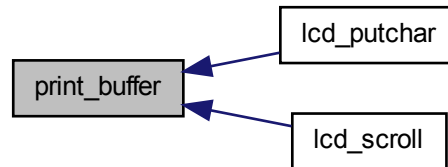
prints the contents of the buffer on the screen. This function is called automatically by `lcd_putchar` when a is received.

Definition at line 72 of file lcd.c.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.4 void push_char (uint8_t data)

push a character into the buffer.

This is a internal function, that should not be called from other files. To print the buffer to the display call [print_buffer\(\)](#)

Parameters

<i>data</i>	Character to print.
-------------	---------------------

Returns

void

Definition at line 163 of file `lcd.c`.

Here is the caller graph for this function:



6.1.3 Variable Documentation

6.1.3.1 char input_buffer[LCD_ROWS][INPUT_BUFFER_SIZE+1]

Initial value:

```
{
    "*****\0",
    "**  UART-Display  **\0",
    "**by Ernst Abrazzo**\0",
    "*****\0"
}
```

buffer used to store the received characters and rotate the lines

Definition at line 27 of file lcd.c.

6.1.3.2 char remap[256]

Initial value:

```
{
    0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14
, 15,
    16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
, 31,
    32, 33, 34, 35, 162, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46
, 47,
    48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62
, 63,
    160, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78
, 79,
    80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 250, 251, 252, 29
, 196,
    96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110
, 111,
    112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 253, 254, 255, 206
, 127,
    128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142
, 143,
    144, 145, 146, 147, 148, 149, 45, 45, 152, 153, 154, 155, 156, 157, 158
, 159,
    160, 40, 162, 64, 164, 163, 254, 95, 168, 169, 170, 171, 172, 173, 174
, 175,
    128, 140, 130, 131, 180, 143, 182, 183, 184, 185, 186, 187, 139, 138, 190
, 96,
    192, 193, 194, 195, 91, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206
, 207,
    208, 209, 210, 211, 212, 213, 92, 215, 216, 217, 218, 219, 94, 221, 222
, 190,
    127, 225, 226, 227, 123, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238
, 239,
    240, 125, 242, 243, 244, 245, 124, 247, 248, 249, 250, 251, 126, 253, 254
, 255
}
```

lookup table to translate display charset to ASCII charset

Definition at line 53 of file lcd.c.

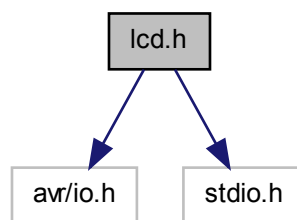
6.2 lcd.h File Reference

Dotmatrix LCD Library.

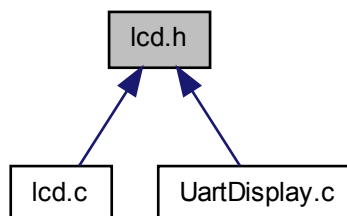
```
#include <avr/io.h>
```

```
#include <stdio.h>
```

Include dependency graph for lcd.h:



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



Defines

- #define `INPUT_BUFFER_SIZE` 60
size of the input buffer used when scrolling is enabled
- #define `LCD_D_PORT` PORTC
Port for Data Lines DB4-DB7 = Px0-Px3.

- #define **LCD_E_PORT** PORTD
Port for Control Line E.
- #define **LCD_RS_PORT** PORTD
Port for Control Line RS.
- #define **LCD_RW_PORT** PORTD
Port for Control Line RW.
- #define **LCD_DATAMSK** 0x0f
Mask of the Data Pins.
- #define **LCD_RS** 4
Register Set Pin Bitposition.
- #define **LCD_E** 2
LCD Enable Pin Bitposition.
- #define **LCD_RW** 3
Read / Write Pin Bitposition.
- #define **LCD_TYPE** 3
- #define **ks0073**
if the controller is a KS0073 the addresses of the LDC-Rows are slightly different
- #define **ENTRY_MODE** 0b00000100
- #define **DISPLAY_STATUS** 0b00001100
- #define **FUNCTION_SET** 0b00101000
- #define **FUNCTION_SET_RE** 0b00100100
- #define **EXT_FUNCTION_SET** 0b00001001
- #define **LCD_CMD_CLEAR** 0x01
LCD Command: Clear Screen.
- #define **LCD_CMD_HOME** 0x02
LCD Command: move cursor to upper left corner.
- #define **LCD_CMD_MOVE_RIGHT** 0x14
LCD Command: move cursor right by one position.
- #define **LCD_CMD_MOVE_LEFT** 0x10
LCD Command: move cursor left by one position.
- #define **LCD_CMD_SHIFT_RIGHT** 0x1C
LCD Command: shift right.

- #define `LCD_CMD_SHIFT_LEFT` 0x18
LCD Command: shift left.
- #define `LCD_CMD_DISPLAY_OFF` 0x08
LCD Command: Display off.
- #define `LCD_CMD_DISPLAY_ON` 0x0C
LCD Command: Display on (CUESOR and BLINK is OFF)
- #define `LCD_CMD_CURSOR_ON` 0x0E
LCD Command: Cursor on.
- #define `LCD_CMD_BLINK_ON` 0x0D
LCD Command: Blink on.
- #define `lcd_home()` `lcd_command(LCD_CMD_HOME);`
move the Cursor to to the upper left corner of the screen.
- #define `lcd_clear()` `lcd_command(LCD_CMD_CLEAR);`
sends the command clear the display
- #define `LCD_D_DDR` (`LCD_D_PORT-1`)
- #define `LCD_D_PIN` (`LCD_D_PORT-2`)
- #define `LCD_E_DDR` (`LCD_E_PORT-1`)
- #define `LCD_RS_DDR` (`LCD_RS_PORT-1`)
- #define `LCD_RW_DDR` (`LCD_RW_PORT-1`)
- #define `LCD_ROWS` 4
- #define `LCD_COLS` 20
- #define `LCD_LINE1` 0x00
- #define `LCD_LINE2` 0x20
- #define `LCD_LINE3` 0x40
- #define `LCD_LINE4` 0x60

Functions

- void `lcd_data` (`uint8_t data`)
send one data byte
- void `lcd_command` (`uint8_t cmd`)
send a command byte
- void `lcd_setadr` (`uint8_t adr`)
set a specific display address
- void `lcd_puts` (`char const *c`)
write a string

- void [lcd_puts_P](#) (char const *)
write a string from flash
- void [lcd_init](#) (void)
initialise the display
- uint8_t [lcd_readadr](#) (void)
read the position of the cursor
- int [lcd_putchar](#) (char data, FILE *sream)
Write one Character on the Display.
- void [print_buffer](#) (void)
send the buffer contents to the display
- void [lcd_scroll](#) (void)
scroll individual lines

6.2.1 Detailed Description

Dotmatrix LCD Library.

Authors

Dominic Rathje (dominic.rathje@uni-ulm.de)
Christian Degenhart (christian.degenhart@uni-ulm.de)

Version

1.0

Note

- Compiler : WinAVR 20100110
- Supported devices : ATmega8

Definition in file [lcd.h](#).

6.2.2 Define Documentation

6.2.2.1 #define LCD_TYPE 3

defines the format of the connected Display 0 = 16*2 1 = 16*4 2 = 40*2 3 = 4*20

Definition at line 34 of file lcd.h.

6.2.3 Function Documentation

6.2.3.1 void lcd_command (uint8_t cmd)

send a command byte

Parameters

<i>cmd</i>	Command to send
------------	-----------------

6.2.3.2 void lcd_data (uint8_t data)

send one data byte

Parameters

<i>data</i>	Byte to send
-------------	--------------

Returns

void

6.2.3.3 void lcd_init (void)

initialise the display

executes the initialisation sequence defined by the HD44780 datasheet

Here is the caller graph for this function:



6.2.3.4 int lcd_putchar (char data, FILE * stream)

Write one Character on the Display.

Characters received are transformer by a lookup-table because the charset of the Display is not equal to ASCII. As input to this function any ASCII character that can be typed ba a German or American Keyboard is valid and displayed correctly on the screen.

As the Display has some amazing special characters, like fancy arrows, there are some sequences that are parsed and replaced by the corresponding symbol. See <https://wiki.fs-et.de/student-lab/UartDisplay> for details on the parser.

Parameters

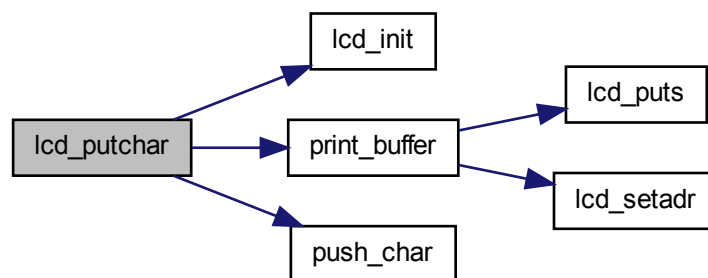
<i>data</i>	ASCII Character to print.
<i>stream</i>	(is necessary to be compatible with printf)

Returns

0 on success.

Definition at line 181 of file lcd.c.

Here is the call graph for this function:

**6.2.3.5 void lcd_puts (char const * c)**

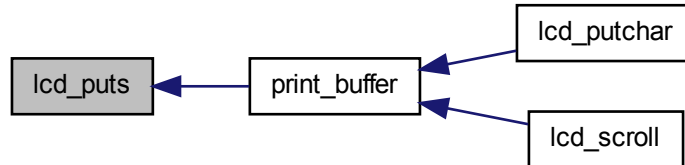
write a string

writes a sequence of characters until the first zero-byte.

Parameters

<i>c</i>	Pointer to a terminated string
----------	--------------------------------

Here is the caller graph for this function:



6.2.3.6 void lcd_puts_P (char const *)

write a string from flash

same as `lcd_puts` expect that the string is located in flash memory.

See also

[lcd_puts](#)

Parameters

<code>c</code>	Pointer to a terminated string
----------------	--------------------------------

6.2.3.7 uint8_t lcd_readadr (void)

read the position of the cursor

Returns

current address of the cursor position

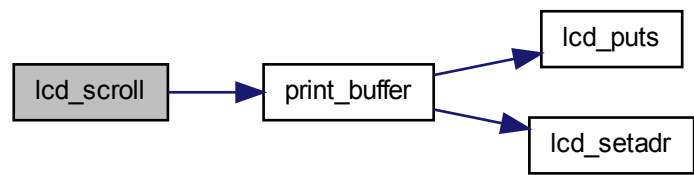
6.2.3.8 void lcd_scroll (void)

scroll individual lines

This function should be called by the main loop regularly if scroll mode is activated. Each time this function is executed the content of a line longer than the physical display length is rotated by one position.

Definition at line 128 of file `lcd.c`.

Here is the call graph for this function:



6.2.3.9 void lcd_setadr (uint8_t adr)

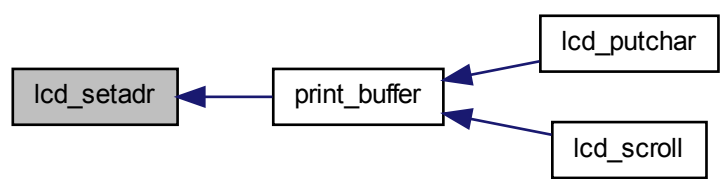
set a specific display address

Moves the cursor to a specific address, so the next character sent will be displayed at this position. To get the address of a certain position use the offset of the line (LCD_LINE[1..4]) and add the desired column.

Parameters

<i>adr</i>	Address
------------	---------

Here is the caller graph for this function:



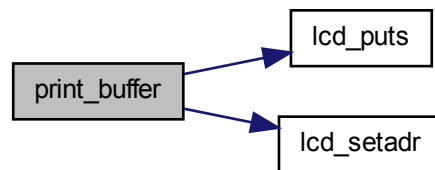
6.2.3.10 void print_buffer (void)

send the buffer contents to the display

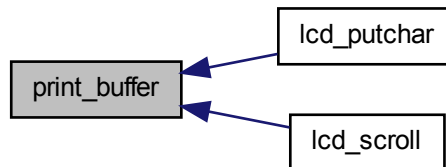
prints the contents of the buffer on the screen. This function is called automatically by `lcd_putchar` when a is received.

Definition at line 72 of file `lcd.c`.

Here is the call graph for this function:



Here is the caller graph for this function:

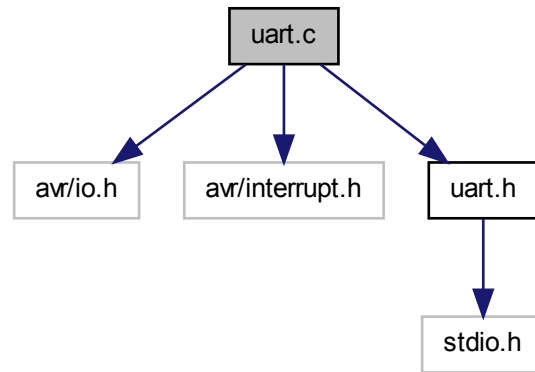


6.3 uart.c File Reference

Implementation of USART Stack.

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

Include dependency graph for uart.c:



Defines

- `#define _UART_C_ 1`

Functions

- void `uart_init` (void)
Initialise the USART Module.
- void `uart_sendbyte` (uint8_t data)
Send one Byte.
- uint8_t `uart_senddata` (uint8_t data)
Send one data byte without buffering in the FIFO.
- uint8_t `uart_sendid` (uint8_t data)
Send one id byte without buffering in the FIFO.
- uint8_t `uart_getbyte` (uint8_t *data)
Pull one Byte from RX-Buffer.
- int `uart_putchar` (char c, FILE *sream)
Send one Byte (printf version)

- `int uart_getchar (FILE *stream)`
Pull one Byte from RX-Buffer.
- `ISR (RXC_vect)`
Interrupt-Service-Routine for the RX-Complete Interrupt.
- `ISR (DRE_vect)`
Interrupt-Service-Routine for the Data-Register-Empty Interrupt.

Variables

- `uint8_t rx_buffer [UART_RX_BUFFER_SIZE]`
Memory for the Receive FIFO.
- `uint8_t tx_buffer [UART_TX_BUFFER_SIZE]`
Memory for the Transmit FIFO.
- `volatile uint8_t rx_head = 0`
head index of the Receive FIFO
- `volatile uint8_t rx_tail = 0`
tail index of the Receive FIFO
- `volatile uint8_t tx_head = 0`
head index of the Transmit FIFO
- `volatile uint8_t tx_tail = 0`
tail index of the Transmit FIFO

6.3.1 Detailed Description

Implementation of USART Stack.

Author

Dominic Rathje (dominic.rathje@uni-ulm.de)

Version

1.0

Note

- Compiler : WinAVR 20100110
- Supported devices : ATMega8

Definition in file [uart.c](#).

6.3.2 Function Documentation

6.3.2.1 ISR (RXC_vect)

Interrupt-Service-Routine for the RX-Complete Interrupt.

If a RX-Complete Interrupt occurs:

- check if there is any space left in the FIFO Buffer
- copy the received byte into the buffer

Definition at line 130 of file uart.c.

6.3.2.2 ISR (DRE_vect)

Interrupt-Service-Routine for the Data-Register-Empty Interrupt.

If a Data-Register-Empty Interrupt occurs:

- check if there is pending data in the FIFO
 - if yes: send the next byte
 - if no: disable URDE Interrupt

Definition at line 145 of file uart.c.

Here is the call graph for this function:



6.3.2.3 uint8_t uart_getbyte (uint8_t * data)

Pull one Byte from RX-Buffer.

If the RX-Fifo is enabled any data receives is automatically pushed into the Fifo by the ISR. This function removes the oldest Byte from the Buffer and copies it into the Memory-Space which is handed over as a parameter.

See also

[uart_getchar](#) for the scanf compatible version

Parameters

<i>data</i>	pointer to a buffer where the received data is copied to
-------------	--

Returns

true if data has been copied. false if the RX buffer was empty

Definition at line 72 of file uart.c.

6.3.2.4 int uart_getchar (FILE * *stream*)

Pull one Byte from RX-Buffer.

If the RX-Fifo is enabled any data receives is automatically pushed into the Fifo by the ISR. This function removes the oldest Byte from the Buffer and copies it into the Memory-Space which is handed over as a parameter.

See also

[uart_getbyte](#) for the standard version

Returns

received data or _FDEV_EOF

Definition at line 110 of file uart.c.

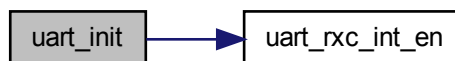
6.3.2.5 void uart_init (void)

Initialise the USART Module.

Configures the Port Registers and the USART Module. Call this function at the beginning of your main function.

Definition at line 27 of file uart.c.

Here is the call graph for this function:



6.3.2.6 int uart_putchar (char *c*, FILE * *stream*)

Send one Byte (printf version)

This fuction sends one byte by copyint the data into the fifo-buffer and enabling the UDRE-Interrupt. The actual transmission is then initiated by the Interrupt-Service-Routine. If the buffer is full this function waits until the next byte has been transmitted, so it is always successful.

This is the printf compatible version ov `uart_sendbyte`

See also

[uart_sendbyte](#) for the standars version

Parameters

<i>data</i>	Data-Byte to send
-------------	-------------------

Returns

always 0

Definition at line 101 of file `uart.c`.

Here is the call graph for this function:



6.3.2.7 void uart_sendbyte (uint8_t *data*)

Send one Byte.

This fuction sends one byte by copying the data into the fifo-buffer and enabling the UDRE-Interrupt. The actual transmission is then initiated by the Interrupt-Service-Routine. If the buffer is full this function waits until the next byte has been transmitted, so it is always successful.

See also

[uart_putchar](#) for the printf compatible version

Parameters

<i>data</i>	Data-Byte to send
-------------	-------------------

Definition at line 48 of file uart.c.

Here is the call graph for this function:



6.3.2.8 uint8_t uart_senddata (uint8_t *data*)

Send one data byte without buffering in the FIFO.

Parameters

<i>data</i>	Data-Byte to send
-------------	-------------------

Returns

true if data has been send. else false

Definition at line 56 of file uart.c.

6.3.2.9 uint8_t uart_sendid (uint8_t *data*)

Send one id byte without buffering in the FIFO.

This function is only for use in 9-Bit Character Size and MPCM Mode. The ninth bit es set to 1 to send an address/id frame.

Parameters

<i>data</i>	Data-Byte to send
-------------	-------------------

Returns

true if data has been send. else false

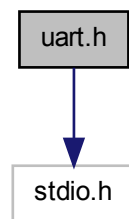
Definition at line 64 of file uart.c.

6.4 uart.h File Reference

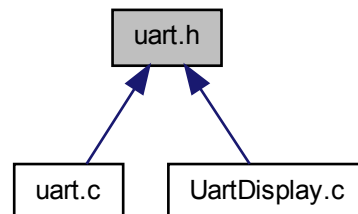
Implementation of USART Stack.

```
#include <stdio.h>
```

Include dependency graph for uart.h:



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



Defines

- #define `UART_DDR` `DDRD`
Data Directory Register of the desired USART.
- #define `RXPIN_bm` `(1<<2)`
Bitmask of the RX-Pin.

- #define `TXPIN_bm` (1<<3)
Bitmask of the TX-Pin.
- #define `UART_DATA` UDR
USART Data Register.
- #define `RXC_vect` USART_RXC_vect
RX-Complete Interrupt Vector.
- #define `DRE_vect` USART_UDRE_vect
Data-Register-Empty Interrupt Vector.
- #define `UART_RX_BUFFER_SIZE` 64
Size of the receiver FIFO.
- #define `UART_TX_BUFFER_SIZE` 64
Size of the transmitter FIFO.
- #define `UBRR_val` 3
Value of Baudrate Register.
- #define `UART_CHRSIZE_9BIT` 0
set Character Size to 9Bit
- #define `UART_PRINTF_COMPATIBILITY` 1
enable printf compatible functions
- #define `UART_MPCM_ENABLE` 0
enable mpcm functions

Functions

- void `uart_init` (void)
Initialise the USART Module.
- void `uart_sendbyte` (uint8_t data)
Send one Byte.
- uint8_t `uart_getbyte` (uint8_t *data)
Pull one Byte from RX-Buffer.
- uint8_t `uart_senddata` (uint8_t data)
Send one data byte without buffering in the FIFO.
- uint8_t `uart_sendid` (uint8_t data)

Send one id byte without buffering in the FIFO.

- void [uart_mpcm_en](#) (void)
Enable Multiprocessor Communication Mode.
- void [uart_mpcm_dis](#) (void)
Disable Multiprocessor Communication Mode.
- void [uart_udre_int_en](#) (void)
Enable Data Register Empty Interrupt.
- void [uart_udre_int_dis](#) (void)
Disable Data Register Empty Interrupt.
- void [uart_rxc_int_en](#) (void)
Enable Data RX Complete Interrupt.
- void [uart_rxc_int_dis](#) (void)
Disable Data RX Complete Interrupt.
- int [uart_putchar](#) (char c, FILE *sream)
Send one Byte (printf version)
- int [uart_getchar](#) (FILE *stream)
Pull one Byte from RX-Buffer.

6.4.1 Detailed Description

Implementation of USART Stack.

Author

Dominic Rathje (dominic.rathje@uni-ulm.de)

Version

1.0

Note

- Compiler : WinAVR 20100110
- Supported devices : ATmega8

Definition in file [uart.h](#).

6.4.2 Define Documentation

6.4.2.1 `#define UART_CHRSIZE_9BIT 0`

set Character Size to 9Bit

Set this to 1 to use 9Bit Character Size. Default is 8Bit

Definition at line 85 of file uart.h.

6.4.2.2 `#define UART_MPCM_ENABLE 0`

enable mpcm functions

In order to save codespace you can set this define to 0 if you do not need the Multi-Processor-Communication-Mode features.. This will diable the functions [uart_senddata\(\)](#) and [uart_sendid\(\)](#) The value 1 enables the Feature

Definition at line 100 of file uart.h.

6.4.2.3 `#define UART_PRINTF_COMPATIBILITY 1`

enable printf compatible functions

In order to save codespace you can set this define to 0 if you do not need printf compatibility. The value 1 enables the Feature

Definition at line 92 of file uart.h.

6.4.2.4 `#define UART_RX_BUFFER_SIZE 64`

Size of the receiver FIFO.

the size must be a power of 2 and smaller or equal to 256. So possible Values are: 2,4,8,16,32,64,128,256

Definition at line 56 of file uart.h.

6.4.2.5 `#define UART_TX_BUFFER_SIZE 64`

Size of the transmitter FIFO.

the size must be a power of 2 and smaller or equal to 256. So possible Values are: 2,4,8,16,32,64,128,256

Definition at line 62 of file uart.h.

6.4.2.6 `#define UBRR_val 3`

Value of Baudrate Register.

Equation for Calculating the Baud Rate Register Setting:

- $UBRR = (f_OSC / (16 * BAUD)) - 1$

example values vor F_CPU=7.3728MHz

- 3 => 115200 BAUD
- 11 => 38400 BAUD
- 47 => 9600 BAUD example values vor F_CPU=14.7456MHz
- 7 => 115200 BAUD
- 23 => 38400 BAUD
- 95 => 9600 BAUD

Definition at line 79 of file uart.h.

6.4.3 Function Documentation

6.4.3.1 `uint8_t uart_getbyte (uint8_t * data)`

Pull one Byte from RX-Buffer.

If the RX-Fifo is enabled any data receives is automaticaly pusched into the Fifo ba the ISR. This function removes the oldest Byte from the Buffer and copies it into the Memory-Space which is handed over as a parameter.

See also

[uart_getchar](#) for the scanf compatible version

Parameters

<code>data</code>	pointer to a buffer where the received data is copied to
-------------------	--

Returns

true if data has been copied. false if the RX buffer was empty

Definition at line 72 of file uart.c.

6.4.3.2 `int uart_getchar (FILE * stream)`

Pull one Byte from RX-Buffer.

If the RX-Fifo is enabled any data receives is automaticaly pusched into the Fifo ba the ISR. This function removes the oldest Byte from the Buffer and copies it into the Memory-Space which is handed over as a parameter.

See also

[uart_getbyte](#) for the standard version

Returns

received data or `_FDEV_EOF`

Definition at line 110 of file `uart.c`.

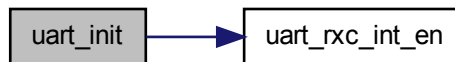
6.4.3.3 void uart_init (void)

Initialise the USART Module.

Configures the Port Registers and the USART Module. Call this function at the beginning of your main function.

Definition at line 27 of file `uart.c`.

Here is the call graph for this function:

**6.4.3.4 int uart_putchar (char c, FILE * stream)**

Send one Byte (printf version)

This function sends one byte by copying the data into the fifo-buffer and enabling the UDRE-Interrupt. The actual transmission is then initiated by the Interrupt-Service-Routine. If the buffer is full this function waits until the next byte has been transmitted, so it is always successful.

This is the printf compatible version of `uart_sendbyte`

See also

[uart_sendbyte](#) for the standard version

Parameters

<i>data</i>	Data-Byte to send
-------------	-------------------

Returns

always 0

Definition at line 101 of file uart.c.

Here is the call graph for this function:

**6.4.3.5 void uart_sendbyte (uint8_t data)**

Send one Byte.

This function sends one byte by copying the data into the fifo-buffer and enabling the UDRE-Interrupt. The actual transmission is then initiated by the Interrupt-Service-Routine. If the buffer is full this function waits until the next byte has been transmitted, so it is always successful.

See also

[uart_putchar](#) for the printf compatible version

Parameters

<i>data</i>	Data-Byte to send
-------------	-------------------

Definition at line 48 of file uart.c.

Here is the call graph for this function:



6.4.3.6 `uint8_t uart_senddata (uint8_t data)`

Send one data byte without buffering in the FIFO.

Parameters

<i>data</i>	Data-Byte to send
-------------	-------------------

Returns

true if data has been send. else false

Definition at line 56 of file uart.c.

6.4.3.7 `uint8_t uart_sendid (uint8_t data)`

Send one id byte without buffering in the FIFO.

This function is only for use in 9-Bit Character Size and MPCM Mode. The ninth bit es set to 1 to send an address/id frame.

Parameters

<i>data</i>	Data-Byte to send
-------------	-------------------

Returns

true if data has been send. else false

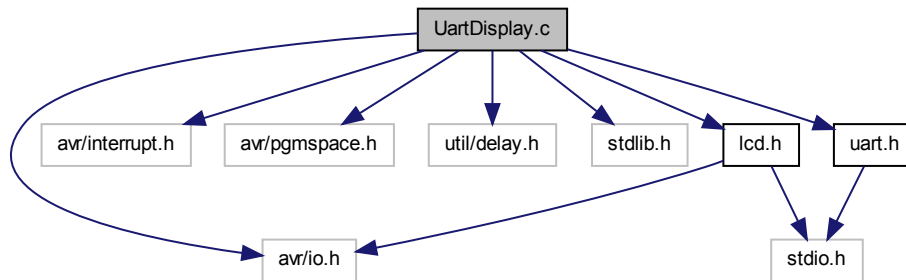
Definition at line 64 of file uart.c.

6.5 UartDisplay.c File Reference

UartDisplay Main File.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include <util/delay.h>
#include <stdlib.h>
#include "lcd.h"
#include "uart.h"
```

Include dependency graph for UartDisplay.c:



Functions

- void `timer2_init` (void)
Initializes timer2.
- int `main` (void)
- ISR (TIMER2_OVF_vect)

Variables

- uint8_t `cnt` = 0
- uint8_t `scroll`
scroll enable flag

6.5.1 Detailed Description

UartDisplay Main File.

Authors

Dominic Rathje (dominic.rathje@uni-ulm.de)
Christian Degenhart (christian.degenhart@uni-ulm.de)

Version

1.0

Note

- Compiler : WinAVR 20100110

- Supported devices : ATmega8

Todo

- Implement Buttons
- implement Buzzer

Definition in file [UartDisplay.c](#).

6.5.2 Function Documentation

6.5.2.1 `void timer2_init (void)` `[inline]`

Initializes timer2.

- Prescaler is set to 1024
- Overflow Interrupt enabled.

Definition at line 49 of file UartDisplay.c.

Index

- input_buffer
 - lcd.c, [17](#)
- ISR
 - uart.c, [29](#)
- lcd.c, [11](#)
 - input_buffer, [17](#)
 - lcd_putchar, [14](#)
 - lcd_scroll, [14](#)
 - print_buffer, [15](#)
 - push_char, [16](#)
 - remap, [17](#)
- lcd.h, [18](#)
 - lcd_command, [22](#)
 - lcd_data, [22](#)
 - lcd_init, [22](#)
 - lcd_putchar, [22](#)
 - lcd_puts, [23](#)
 - lcd_puts_P, [24](#)
 - lcd_readadr, [24](#)
 - lcd_scroll, [24](#)
 - lcd_setadr, [25](#)
 - LCD_TYPE, [21](#)
 - print_buffer, [25](#)
- lcd_command
 - lcd.h, [22](#)
- lcd_data
 - lcd.h, [22](#)
- lcd_init
 - lcd.h, [22](#)
- lcd_putchar
 - lcd.c, [14](#)
 - lcd.h, [22](#)
- lcd_puts
 - lcd.h, [23](#)
- lcd_puts_P
 - lcd.h, [24](#)
- lcd_readadr
 - lcd.h, [24](#)
- lcd_scroll
 - lcd.c, [14](#)
 - lcd.h, [24](#)
- lcd_setadr
 - lcd.h, [25](#)
- LCD_TYPE
 - lcd.h, [21](#)
- print_buffer
 - lcd.c, [15](#)
 - lcd.h, [25](#)
- push_char
 - lcd.c, [16](#)
- remap
 - lcd.c, [17](#)
- timer2_init
 - UartDisplay.c, [42](#)
- uart.c, [26](#)
 - ISR, [29](#)
 - uart_getbyte, [29](#)
 - uart_getchar, [30](#)
 - uart_init, [30](#)
 - uart_putchar, [30](#)
 - uart_sendbyte, [31](#)
 - uart_senddata, [32](#)
 - uart_sendid, [32](#)
- uart.h, [33](#)
 - UART_CHRSIZE_9BIT, [36](#)
 - uart_getbyte, [37](#)
 - uart_getchar, [37](#)
 - uart_init, [38](#)
 - UART_MPCM_ENABLE, [36](#)
 - UART_PRINTF_COMPATIBILITY, [36](#)
 - uart_putchar, [38](#)
 - UART_RX_BUFFER_SIZE, [36](#)
 - uart_sendbyte, [39](#)
 - uart_senddata, [39](#)
 - uart_sendid, [40](#)
 - UART_TX_BUFFER_SIZE, [36](#)
 - UBRR_val, [36](#)
- UART_CHRSIZE_9BIT

- uart.h, [36](#)
- uart_getbyte
 - uart.c, [29](#)
 - uart.h, [37](#)
- uart_getchar
 - uart.c, [30](#)
 - uart.h, [37](#)
- uart_init
 - uart.c, [30](#)
 - uart.h, [38](#)
- UART_MPCM_ENABLE
 - uart.h, [36](#)
- UART_PRINTF_COMPATIBILITY
 - uart.h, [36](#)
- uart_putchar
 - uart.c, [30](#)
 - uart.h, [38](#)
- UART_RX_BUFFER_SIZE
 - uart.h, [36](#)
- uart_sendbyte
 - uart.c, [31](#)
 - uart.h, [39](#)
- uart_senddata
 - uart.c, [32](#)
 - uart.h, [39](#)
- uart_sendid
 - uart.c, [32](#)
 - uart.h, [40](#)
- UART_TX_BUFFER_SIZE
 - uart.h, [36](#)
- UartDisplay.c, [40](#)
 - timer2_init, [42](#)
- UBRR_val
 - uart.h, [36](#)