avr-uart

Generated by Doxygen 1.8.12

Contents

Chapter 1

avr-uart

An interrupt driven UART Library for 8-bit AVR microcontrollers

Maintained by Andy Gock

https://github.com/andygock/avr-uart

Derived from original library by Peter Fleury.

Interrupt driven UART library using the built-in UART with circular transmit and receive buffers.

An interrupt is generated when the UART has finished transmitting or receiving a byte. The interrupt handling routines use circular buffers for buffering received and transmitted data.

Setting up

The `UART_RXn_BUFFER_SIZE` and `UART_TXn_BUFFER_SIZE` symbols define the size of the circular buffers in bytes. These values **must be a power of 2**. You may need to adapt this symbols to your target and your application by adding into your compiler options:

```
-DUART_RXn_BUFFER_SIZE=nn -DUART_TXn_BUFFER_SIZE=nn
```

`RXn` and `TXn` refer to the UART number, for UART3 with 128 byte buffers, add:

```
-DUART_RX3_BUFFER_SIZE=128 -DUART_TX3_BUFFER_SIZE=128
```

UART0 is always enabled by default, to enable the other available UARTs, add the following to your compiler's symbol options for the relevant UART (also known as USART) number.

```
-DUSART1_ENABLED -DUSART2_ENABLED -DUSART3_ENABLED
```

To enable large buffer support (over 256 bytes, up to 2^16 bytes) use:

```
-DUSARTn_LARGE_BUFFER
```

Where n = USART number. The maximum buffer size is 32768.

This library supports AVR devices with up to 4 hardware USARTs.

Compiler flags

AVR/GNU C compiler requires the `-std=gnu99` flag.

Documentation

Doxygen based documentation can be viwed at:

- HTML: https://andygock.github.io/avr-uart/docs/html/
- PDF: https://andygock.github.io/avr-uart/docs/html/
- RTF: https://andygock.github.io/avr-uart/docs/html/

Notes

Buffer overflow behaviour

When the RX circular buffer is full, and it receives further data from the UART, a buffer overflow condition occurs. Any new data is dropped. The RX buffer must be read before any more incoming data from the UART is placed into the RX buffer.

If the TX buffer is full, and new data is sent to it using one of the `uartN_put*()` functions, this function will loop and wait until the buffer is not full any more. It is important to make sure you have not disabled your UART transmit interrupts (`TXEN*`) elsewhere in your application (e.g with `cli()`) before calling the `uartN_put*()` functions, as the application will lock up. The UART interrupts are automatically enabled when you use the `uartN_init()` functions. This is probably not the idea behaviour, I'll probably fix this some time.

For now, make sure `TXEN*` interrupts are enabled when calling `uartN_put*()` functions. This should not be an issue unless you have code elsewhere purposely turning it off.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Module Documentation

3.1 UART Library

Interrupt UART library using the built-in UART with transmit and receive circular buffers.

Macros

- #define USART0_ENABLED
- #define UART_RX0_BUFFER_SIZE 128
- #define UART_RX1_BUFFER_SIZE 128
- #define UART_RX2_BUFFER_SIZE 128
- #define UART_RX3_BUFFER_SIZE 128
- #define UART_TX0_BUFFER_SIZE 128
- #define UART_TX1_BUFFER_SIZE 128
- #define UART_TX2_BUFFER_SIZE 128
- #define UART_TX3_BUFFER_SIZE 128
- #define UART_BAUD_SELECT(baudRate, xtalCpu) (((xtalCpu)+8UL∗(baudRate))/(16UL∗(baudRate))-1UL)

  UART Baudrate Expression.
- #define UART_BAUD_SELECT_DOUBLE_SPEED(baudRate, xtalCpu) ((((xtalCpu)+4UL∗(baudRate))/(8↩ UL∗(baudRate))-1)|0x8000)

  UART Baudrate Expression for ATmega double speed mode.
- #define UART_FRAME_ERROR 0x0800
- #define UART_OVERRUN_ERROR 0x0400
- #define UART_BUFFER_OVERFLOW 0x0200
- #define UART_NO_DATA 0x0100
- #define uart_init(b) uart0_init(b)

  Macro to initialize USART0 (only available on selected ATmegas)
- #define uart_getc() uart0_getc()

  Macro to get received byte of USART0 from ringbuffer. (only available on selected ATmega)
- #define uart_peek() uart0_peek()

  Macro to peek at next byte in USART0 ringbuffer.
- #define uart_putc(d) uart0_putc(d)

  Macro to put byte to ringbuffer for transmitting via USART0 (only available on selected ATmega)
- #define uart_puts(s) uart0_puts(s)

  Macro to put string to ringbuffer for transmitting via USART0 (only available on selected ATmega)
- #define uart_puts_p(s) uart0_puts_p(s)

Macro to put string from program memory to ringbuffer for transmitting via USART0 (only available on selected ATmega)

- #define uart_available() uart0_available()

  Macro to return number of bytes waiting in the receive buffer of USART0.
- #define uart_flush() uart0_flush()

  Macro to flush bytes waiting in receive buffer of USART0.
- #define uart_puts_P(__s) uart0_puts_p(PSTR(__s))

  Macro to automatically put a string constant into program memory.
- #define uart0_puts_P(__s) uart0_puts_p(PSTR(__s))

  Macro to automatically put a string constant into program memory.
- #define uart1_puts_P(__s) uart1_puts_p(PSTR(__s))

  Macro to automatically put a string constant into program memory of USART1.
- #define uart2_puts_P(__s) uart2_puts_p(PSTR(__s))

  Macro to automatically put a string constant into program memory of USART2.
- #define uart3_puts_P(__s) uart3_puts_p(PSTR(__s))

  Macro to automatically put a string constant into program memory of USART3.

Functions

- void uart0_init (uint16_t baudrate)

  Initialize UART and set baudrate.
- uint16_t uart0_getc (void)

  Get received byte from ringbuffer.
- uint16_t uart0_peek (void)

  Peek at next byte in ringbuffer.
- void uart0_putc (uint8_t data)

  Put byte to ringbuffer for transmitting via UART.
- void uart0_puts (const char *s)

  Put string to ringbuffer for transmitting via UART.
- void uart0_puts_p (const char *s)

  Put string from program memory to ringbuffer for transmitting via UART.
- uint16_t uart0_available (void)

  Return number of bytes waiting in the receive buffer.
- void uart0_flush (void)

  Flush bytes waiting in receive buffer.
- void uart1_init (uint16_t baudrate)

  Initialize USART1 (only available on selected ATmegas)
- uint16_t uart1_getc (void)

  Get received byte of USART1 from ringbuffer. (only available on selected ATmega)
- uint16_t uart1_peek (void)

  Peek at next byte in USART1 ringbuffer.
- void uart1_putc (uint8_t data)

  Put byte to ringbuffer for transmitting via USART1 (only available on selected ATmega)
- void uart1_puts (const char *s)

  Put string to ringbuffer for transmitting via USART1 (only available on selected ATmega)
- void uart1_puts_p (const char *s)

  Put string from program memory to ringbuffer for transmitting via USART1 (only available on selected ATmega)
- uint16_t uart1_available (void)

  Return number of bytes waiting in the receive buffer of USART1.
- void uart1_flush (void)

Flush bytes waiting in receive buffer of USART1.

- void uart2_init (uint16_t baudrate)

  Initialize USART2 (only available on selected ATmegas)
- uint16_t uart2_getc (void)

  Get received byte of USART2 from ringbuffer. (only available on selected ATmega)
- uint16_t uart2_peek (void)

  Peek at next byte in USART2 ringbuffer.
- void uart2_putc (uint8_t data)

  Put byte to ringbuffer for transmitting via USART2 (only available on selected ATmega)
- void uart2_puts (const char ∗s)

  Put string to ringbuffer for transmitting via USART2 (only available on selected ATmega)
- void uart2_puts_p (const char ∗s)

  Put string from program memory to ringbuffer for transmitting via USART2 (only available on selected ATmega)
- uint16_t uart2_available (void)

  Return number of bytes waiting in the receive buffer of USART2.
- void uart2_flush (void)

  Flush bytes waiting in receive buffer of USART2.
- void uart3_init (uint16_t baudrate)

  Initialize USART3 (only available on selected ATmegas)
- uint16_t uart3_getc (void)

  Get received byte of USART3 from ringbuffer. (only available on selected ATmega)
- uint16_t uart3_peek (void)

  Peek at next byte in USART3 ringbuffer.
- void uart3_putc (uint8_t data)

  Put byte to ringbuffer for transmitting via USART3 (only available on selected ATmega)
- void uart3_puts (const char ∗s)

  Put string to ringbuffer for transmitting via USART3 (only available on selected ATmega)
- void uart3_puts_p (const char ∗s)

  Put string from program memory to ringbuffer for transmitting via USART3 (only available on selected ATmega)
- uint16_t uart3_available (void)

  Return number of bytes waiting in the receive buffer of USART3.
- void uart3_flush (void)

  Flush bytes waiting in receive buffer of USART3.

3.1.1 Detailed Description

Interrupt UART library using the built-in UART with transmit and receive circular buffers.

```
#include <uart.h>
```

See also

README.md

This library can be used to transmit and receive data through the built in UART.

An interrupt is generated when the UART has finished transmitting or receiving a byte. The interrupt handling routines use circular buffers for buffering received and transmitted data.

The UART_RXn_BUFFER_SIZE and UART_TXn_BUFFER_SIZE constants define the size of the circular buffers in bytes. Note that these constants must be a power of 2.

You need to define these buffer sizes as a symbol in your compiler settings or in uart.h

See README.md for more detailed information. Especially that relating to symbols: USARTn_ENABLED and USARTn_LARGE_BUFFER

Author

Andy Gock andy@gock.net

Note

Based on Atmel Application Note AVR306 and original library by Peter Fleury and Tim Sharpe.

3.1.2 Macro Definition Documentation

3.1.2.1 uart1_puts_P

```
#define uart1_puts_P(
              __s ) uart1_puts_p(PSTR(__s))
```

Macro to automatically put a string constant into program memory of USART1.

See also

uart1_puts_p

Definition at line 369 of file uart.h.

3.1.2.2 uart2_puts_P

```
#define uart2_puts_P(
              __s ) uart2_puts_p(PSTR(__s))
```

Macro to automatically put a string constant into program memory of USART2.

See also

uart2_puts_p

Definition at line 397 of file uart.h.

3.1.2.3 uart3_puts_P

```
#define uart3_puts_P(
              __s ) uart3_puts_p(PSTR(__s))
```

Macro to automatically put a string constant into program memory of USART3.

See also

uart3_puts_p

Definition at line 425 of file uart.h.

3.1.2.4 uart_available

```
#define uart_available( ) uart0_available()
```

Macro to return number of bytes waiting in the receive buffer of USART0.

See also

uart0_available

Definition at line 224 of file uart.h.

3.1.2.5 UART_BAUD_SELECT

```
#define UART_BAUD_SELECT(
              baudRate,
              xtalCpu ) (((xtalCpu)+8UL*(baudRate))/(16UL*(baudRate))-1UL)
```

UART Baudrate Expression.

Parameters

xtalCpu	system clock in Mhz, e.g. 4000000L for 4Mhz
baudRate	baudrate in bps, e.g. 1200, 2400, 9600

Definition at line 169 of file uart.h.

3.1.2.6 UART_BAUD_SELECT_DOUBLE_SPEED

```
#define UART_BAUD_SELECT_DOUBLE_SPEED(
              baudRate,
              xtalCpu ) ((((xtalCpu)+4UL*(baudRate))/(8UL*(baudRate))-1)|0x8000)
```

UART Baudrate Expression for ATmega double speed mode.

Parameters

xtalCpu	system clock in Mhz, e.g. 4000000L for 4Mhz
baudRate	baudrate in bps, e.g. 1200, 2400, 9600

Definition at line 175 of file uart.h.

3.1.2.7 UART_BUFFER_OVERFLOW

```
#define UART_BUFFER_OVERFLOW 0x0200
```

receive ringbuffer overflow

Definition at line 200 of file uart.h.

3.1.2.8 uart_flush

```
#define uart_flush( ) uart0_flush()
```

Macro to flush bytes waiting in receive buffer of USART0.

See also

uart0_flush

Definition at line 227 of file uart.h.

3.1.2.9 UART_FRAME_ERROR

```
#define UART_FRAME_ERROR 0x0800
```

Framing Error by UART

Definition at line 198 of file uart.h.

3.1.2.10 uart_getc

```
#define uart_getc( ) uart0_getc()
```

Macro to get received byte of USART0 from ringbuffer. (only available on selected ATmega)

See also

uart0_getc

Definition at line 209 of file uart.h.

3.1.2.11 uart_init

```
#define uart_init(
              b ) uart0_init(b)
```

Macro to initialize USART0 (only available on selected ATmegas)

See also

uart0_init

Definition at line 206 of file uart.h.

3.1.2.12 UART_NO_DATA

```
#define UART_NO_DATA 0x0100
```

no receive data available

Definition at line 201 of file uart.h.

3.1.2.13 UART_OVERRUN_ERROR

```
#define UART_OVERRUN_ERROR 0x0400
```

Overrun condition by UART

Definition at line 199 of file uart.h.

3.1.2.14 uart_putc

```
#define uart_putc(
              d ) uart0_putc(d)
```

Macro to put byte to ringbuffer for transmitting via USART0 (only available on selected ATmega)

See also

uart0_putc

Definition at line 215 of file uart.h.

3.1.2.15 uart_puts

```
#define uart_puts(
              s ) uart0_puts(s)
```

Macro to put string to ringbuffer for transmitting via USART0 (only available on selected ATmega)

See also

uart0_puts

Definition at line 218 of file uart.h.

3.1.2.16 uart_puts_p

```
#define uart_puts_p(
              s ) uart0_puts_p(s)
```

Macro to put string from program memory to ringbuffer for transmitting via USART0 (only available on selected ATmega)

See also

uart0_puts_p

Definition at line 221 of file uart.h.

3.1.2.17 uart_puts_P

```
#define uart_puts_P(
              __s ) uart0_puts_p(PSTR(__s))
```

Macro to automatically put a string constant into program memory.

Parameters

_↩ _s	string in program memory

Definition at line 333 of file uart.h.

3.1.2.18 UART_RX0_BUFFER_SIZE

```
#define UART_RX0_BUFFER_SIZE 128
```

Size of the circular receive buffer, must be power of 2

Definition at line 104 of file uart.h.

3.1.2.19 UART_RX1_BUFFER_SIZE

```
#define UART_RX1_BUFFER_SIZE 128
```

Size of the circular receive buffer, must be power of 2

Definition at line 107 of file uart.h.

3.1.2.20 UART_RX2_BUFFER_SIZE

```
#define UART_RX2_BUFFER_SIZE 128
```

Size of the circular receive buffer, must be power of 2

Definition at line 110 of file uart.h.

3.1.2.21 UART_RX3_BUFFER_SIZE

```
#define UART_RX3_BUFFER_SIZE 128
```

Size of the circular receive buffer, must be power of 2

Definition at line 113 of file uart.h.

3.1.2.22 UART_TX0_BUFFER_SIZE

```
#define UART_TX0_BUFFER_SIZE 128
```

Size of the circular transmit buffer, must be power of 2

Definition at line 117 of file uart.h.

3.1.2.23 UART_TX1_BUFFER_SIZE

```
#define UART_TX1_BUFFER_SIZE 128
```

Size of the circular transmit buffer, must be power of 2

Definition at line 120 of file uart.h.

3.1.2.24 UART_TX2_BUFFER_SIZE

```
#define UART_TX2_BUFFER_SIZE 128
```

Size of the circular transmit buffer, must be power of 2

Definition at line 123 of file uart.h.

3.1.2.25 UART_TX3_BUFFER_SIZE

```
#define UART_TX3_BUFFER_SIZE 128
```

Size of the circular transmit buffer, must be power of 2

Definition at line 126 of file uart.h.

3.1.2.26 USART0_ENABLED

```
#define USART0_ENABLED
```

Enable USART0

Definition at line 95 of file uart.h.

3.1.3 Function Documentation

3.1.3.1 uart0_available()

```
uint16_t uart0_available (
            void  )
```

Return number of bytes waiting in the receive buffer.

Returns

bytes waiting in the receive buffer

3.1.3.2 uart0_getc()

```
uint16_t uart0_getc (
            void  )
```

Get received byte from ringbuffer.

Returns in the lower byte the received character and in the higher byte the last receive error. UART_NO_DATA is returned when no data is available.

Returns

lower byte: received byte from ringbuffer
higher byte: last receive status

- **0** successfully received data from UART
- **UART_NO_DATA**
  no receive data available
- **UART_BUFFER_OVERFLOW**
  Receive ringbuffer overflow. We are not reading the receive buffer fast enough, one or more received character have been dropped
- **UART_OVERRUN_ERROR**
  Overrun condition by UART. A character already present in the UART UDR register was not read by the interrupt handler before the next character arrived, one or more received characters have been dropped.
- **UART_FRAME_ERROR**
  Framing Error by UART

3.1.3.3 uart0_init()

```
void uart0_init (
            uint16_t baudrate )
```

Initialize UART and set baudrate.

Parameters

baudrate	Specify baudrate using macro UART_BAUD_SELECT()

Returns

none

3.1.3.4 uart0_peek()

```
uint16_t uart0_peek (
            void  )
```

Peek at next byte in ringbuffer.

Returns the next byte (character) of incoming UART data without removing it from the internal ring buffer. That is, successive calls to uartN_peek() will return the same character, as will the next call to uartN_getc().

UART_NO_DATA is returned when no data is available.

Returns

lower byte: next byte in ringbuffer
higher byte: last receive status

- **0** successfully received data from UART
- **UART_NO_DATA**
  no receive data available
- **UART_BUFFER_OVERFLOW**
  Receive ringbuffer overflow. We are not reading the receive buffer fast enough, one or more received character have been dropped
- **UART_OVERRUN_ERROR**
  Overrun condition by UART. A character already present in the UART UDR register was not read by the interrupt handler before the next character arrived, one or more received characters have been dropped.
- **UART_FRAME_ERROR**
  Framing Error by UART

3.1.3.5 uart0_putc()

```
void uart0_putc (
            uint8_t data )
```

Put byte to ringbuffer for transmitting via UART.

Parameters

data	byte to be transmitted

Returns

none

3.1.3.6 uart0_puts()

```
void uart0_puts (
            const char * s )
```

Put string to ringbuffer for transmitting via UART.

The string is buffered by the uart library in a circular buffer and one character at a time is transmitted to the UART using interrupts. Blocks if it can not write the whole string into the circular buffer.

Parameters

s	string to be transmitted

Returns

none

3.1.3.7 uart0_puts_p()

```
void uart0_puts_p (
            const char * s )
```

Put string from program memory to ringbuffer for transmitting via UART.

The string is buffered by the uart library in a circular buffer and one character at a time is transmitted to the UART using interrupts. Blocks if it can not write the whole string into the circular buffer.

Parameters

s	program memory string to be transmitted

Returns

none

See also

uart0_puts_P

3.1.3.8 uart1_getc()

```
uint16_t uart1_getc (
            void  )
```

Get received byte of USART1 from ringbuffer. (only available on selected ATmega)

See also

uart_getc

3.1.3.9 uart1_init()

```
void uart1_init (
            uint16_t baudrate )
```

Initialize USART1 (only available on selected ATmegas)

See also

uart_init

3.1.3.10 uart1_putc()

```
void uart1_putc (
            uint8_t data )
```

Put byte to ringbuffer for transmitting via USART1 (only available on selected ATmega)

See also

uart_putc

3.1.3.11 uart1_puts()

```
void uart1_puts (
            const char * s )
```

Put string to ringbuffer for transmitting via USART1 (only available on selected ATmega)

See also

uart_puts

3.1.3.12 uart1_puts_p()

```
void uart1_puts_p (
            const char * s )
```

Put string from program memory to ringbuffer for transmitting via USART1 (only available on selected ATmega)

See also

uart_puts_p

3.1.3.13 uart2_getc()

```
uint16_t uart2_getc (
            void  )
```

Get received byte of USART2 from ringbuffer. (only available on selected ATmega)

See also

uart_getc

3.1.3.14 uart2_init()

```
void uart2_init (
            uint16_t baudrate )
```

Initialize USART2 (only available on selected ATmegas)

See also

uart_init

3.1.3.15 uart2_putc()

```
void uart2_putc (
            uint8_t data )
```

Put byte to ringbuffer for transmitting via USART2 (only available on selected ATmega)

See also

uart_putc

3.1.3.16 uart2_puts()

```
void uart2_puts (
            const char * s )
```

Put string to ringbuffer for transmitting via USART2 (only available on selected ATmega)

See also

uart_puts

3.1.3.17 uart2_puts_p()

```
void uart2_puts_p (
            const char * s )
```

Put string from program memory to ringbuffer for transmitting via USART2 (only available on selected ATmega)

See also

uart_puts_p

3.1.3.18 uart3_getc()

```
uint16_t uart3_getc (
            void  )
```

Get received byte of USART3 from ringbuffer. (only available on selected ATmega)

See also

uart_getc

3.1.3.19 uart3_init()

```
void uart3_init (
            uint16_t baudrate )
```

Initialize USART3 (only available on selected ATmegas)

See also

uart_init

3.1.3.20 uart3_putc()

```
void uart3_putc (
            uint8_t data )
```

Put byte to ringbuffer for transmitting via USART3 (only available on selected ATmega)

See also

uart_putc

3.1.3.21 uart3_puts()

```
void uart3_puts (
            const char * s )
```

Put string to ringbuffer for transmitting via USART3 (only available on selected ATmega)

See also

uart_puts

3.1.3.22 uart3_puts_p()

```
void uart3_puts_p (
            const char * s )
```

Put string from program memory to ringbuffer for transmitting via USART3 (only available on selected ATmega)

See also

uart_puts_p

Index