avrSerial

1.0

Generated by Doxygen 1.8.3.1

Sun Apr 28 2013 15:55:39

Contents

Chapter 1

Main Page

This is a serial library for many different Atmel AVR MCUs. It is using two FIFO Buffers per USART module for interrupt driven UART communication. XON/XOFF Flow control for incoming data can be enabled for all UART modules, it will operate independently for each.

Device-specific configuration is in serial_device.h. You should be able to easily add new AVR MCUs. Just get the relevant register and bit names from the data-sheet.

A small test application is included. It will be built when calling either of these commands

```
make all
make test.hex
```

You can also flash it with

```
make program
```

just adjust your programmer type and port in the makefile.

For normal use, either include serial.c, serial.h and serial_device.h in your project, or build a statically-linked library by calling

```
make lib
```

and linking this to your project, as well as including serial.h.

The current Doxygen Documentation can be found in doc/ and on the web or as PDF...

License

Everything is released under a BSD 2-Clause License.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Module Documentation

4.1 UART Library

UART Library enabling you to control all available UART Modules.

Files

- file serial.c

  UART Library Implementation.
- file serial.h

  UART Library Header File.
- file serial_device.h

  UART Library device-specific configuration.

Macros

- #define RX_BUFFER_SIZE 32

  If you define this, a '\r' (CR) will be put in front of a '\n' (LF) when sending a byte.
- #define TX_BUFFER_SIZE 16

  TX Buffer Size in Bytes (Power of 2)
- #define FLOWCONTROL

  Defining this enables incoming XON XOFF (sends XOFF if rx buff is full)
- #define FLOWMARK 5

  Space remaining to trigger xoff/xon.
- #define XON 0x11

  XON Value.
- #define XOFF 0x13

  XOFF Value.
- #define BAUD(baudRate, xtalCpu) ((xtalCpu)/((baudRate)∗16l)-1)

  Calculate Baudrate Register Value.

Functions

- uint8_t serialAvailable (void)

  Get number of available UART modules.
- void serialInit (uint8_t uart, uint16_t baud)

Initialize the UART Hardware.

- void serialClose (uint8_t uart)

  Stop the UART Hardware.
- void setFlow (uint8_t uart, uint8_t on)

  Manually change the flow control.
- uint8_t serialHasChar (uint8_t uart)

  Check if a byte was received.
- uint8_t serialGetBlocking (uint8_t uart)

  Wait until a character is received.
- uint8_t serialGet (uint8_t uart)

  Read a single byte.
- uint8_t serialRxBufferFull (uint8_t uart)

  Check if the receive buffer is full.
- uint8_t serialRxBufferEmpty (uint8_t uart)

  Check if the receive buffer is empty.
- void serialWrite (uint8_t uart, uint8_t data)

  Send a byte.
- void serialWriteString (uint8_t uart, const char ∗data)

  Send a string.
- uint8_t serialTxBufferFull (uint8_t uart)

  Check if the transmit buffer is full.
- uint8_t serialTxBufferEmpty (uint8_t uart)

  Check if the transmit buffer is empty.

4.1.1 Detailed Description

UART Library enabling you to control all available UART Modules. With XON/XOFF Flow Control and buffered Receiving and Transmitting.

4.1.2 Macro Definition Documentation

4.1.2.1 #define BAUD(*baudRate, xtalCpu*) ((xtalCpu)/((baudRate)∗16l)-1)

Calculate Baudrate Register Value.

Examples:

test.c.

Definition at line 45 of file serial.h.

4.1.2.2 #define FLOWCONTROL

Defining this enables incoming XON XOFF (sends XOFF if rx buff is full)

Definition at line 62 of file serial.c.

4.1.2.3 #define FLOWMARK 5

Space remaining to trigger xoff/xon.

Definition at line 64 of file serial.c.

Referenced by serialGet().

4.1.2.4 #define RX_BUFFER_SIZE 32

If you define this, a '\r' (CR) will be put in front of a '\n' (LF) when sending a byte.

Binary Communication will then be impossible!RX Buffer Size in Bytes (Power of 2)

Definition at line 54 of file serial.c.

Referenced by serialGet(), and serialRxBufferFull().

4.1.2.5 #define TX_BUFFER_SIZE 16

TX Buffer Size in Bytes (Power of 2)

Definition at line 58 of file serial.c.

Referenced by serialTxBufferFull(), and serialWrite().

4.1.2.6 #define XOFF 0x13

XOFF Value.

Definition at line 66 of file serial.c.

Referenced by setFlow().

4.1.2.7 #define XON 0x11

XON Value.

Definition at line 65 of file serial.c.

Referenced by serialGet(), and setFlow().

4.1.3 Function Documentation

4.1.3.1 uint8_t serialAvailable (void)

Get number of available UART modules.

Returns

number of modules

Examples:

test.c.

Definition at line 113 of file serial.c.

```
113                                {
114     return UART_COUNT;
115 }
```

4.1.3.2 void serialClose (uint8_t *uart*)

Stop the UART Hardware.

Parameters

uart	UART Module to stop

Definition at line 148 of file serial.c.

References serialTxBufferEmpty().

```
148                                 {
149     if (uart >= UART_COUNT)
150         return;
151 
152     uint8_t sreg = SREG;
153     sei();
154     while (!serialTxBufferEmpty(uart));
155     while (*serialRegisters[uart][SERIALB] & (1 << serialBits[uart][SERIALUDRIE])); // Wait while Transmit
      Interrupt is on
156     cli();
157     *serialRegisters[uart][SERIALB] = 0;
158     *serialRegisters[uart][SERIALC] = 0;
159     SREG = sreg;
160 }
```

4.1.3.3 uint8_t serialGet (uint8_t *uart*)

Read a single byte.

Parameters

uart	UART Module to read from

Returns

Received byte or 0

Examples:

test.c.

Definition at line 217 of file serial.c.

References FLOWMARK, RX_BUFFER_SIZE, and XON.

Referenced by serialGetBlocking().

```
217                                 {
218     if (uart >= UART_COUNT)
219         return 0;
220 
221     uint8_t c;
222 
223 #ifdef FLOWCONTROL
224     rxBufferElements[uart]--;
225     if ((flow[uart] == 0) && (rxBufferElements[uart] <= FLOWMARK)) {
226         while (sendThisNext[uart] != 0);
227         sendThisNext[uart] = XON;
228         flow[uart] = 1;
229         if (shouldStartTransmission[uart]) {
230             shouldStartTransmission[uart] = 0;
231             *serialRegisters[uart][SERIALB] |= (1 << serialBits[uart][SERIALUDRIE]); // Enable Interrupt
232             *serialRegisters[uart][SERIALA] |= (1 << serialBits[uart][SERIALUDRE]); // Trigger Interrupt
233         }
234     }
235 #endif
236 
237     if (rxRead[uart] != rxWrite[uart]) {
238         c = rxBuffer[uart][rxRead[uart]];
239         rxBuffer[uart][rxRead[uart]] = 0;
240         if (rxRead[uart] < (RX_BUFFER_SIZE - 1)) {
241             rxRead[uart]++;
242         } else {
243             rxRead[uart] = 0;
244         }
245         return c;
246     } else {
247         return 0;
248     }
249 }
```

4.1.3.4 uint8_t serialGetBlocking (uint8_t *uart*)

Wait until a character is received.

Parameters

uart	UART Module to read from

Returns

Received byte

Definition at line 209 of file serial.c.

References serialGet(), and serialHasChar().

```
209                                         {
210     if (uart >= UART_COUNT)
211         return 0;
212
213     while(!serialHasChar(uart));
214     return serialGet(uart);
215 }
```

4.1.3.5 uint8_t serialHasChar (uint8_t *uart*)

Check if a byte was received.

Parameters

uart	UART Module to check

Returns

1 if a byte was received, 0 if not

Examples:

test.c.

Definition at line 198 of file serial.c.

Referenced by serialGetBlocking().

```
198                                       {
199     if (uart >= UART_COUNT)
200         return 0;
201
202     if (rxRead[uart] != rxWrite[uart]) { // True if char available
203         return 1;
204     } else {
205         return 0;
206     }
207 }
```

4.1.3.6 void serialInit (uint8_t *uart*, uint16_t *baud*)

Initialize the UART Hardware.

Parameters

uart	UART Module to initialize
baud	Baudrate. Use the BAUD() macro!

Examples:

test.c.

Definition at line 117 of file serial.c.

```
117                                                 {
118     if (uart >= UART_COUNT)
119         return;
120
121     // Initialize state variables
122     rxRead[uart] = 0;
123     rxWrite[uart] = 0;
124     txRead[uart] = 0;
125     txWrite[uart] = 0;
126     shouldStartTransmission[uart] = 1;
127 #ifdef FLOWCONTROL
128     sendThisNext[uart] = 0;
129     flow[uart] = 1;
130     rxBufferElements[uart] = 0;
131 #endif
132
133     // Default Configuration: 8N1
134     *serialRegisters[uart][SERIALC] = (1 << serialBits[uart][SERIALUCSZ0]) | (1 << serialBits[uart][
      SERIALUCSZ1]);
135
136     // Set baudrate
137 #if SERIALBAUDBIT == 8
138     *serialRegisters[uart][SERIALUBRRH] = (baud >> 8);
139     *serialRegisters[uart][SERIALUBRRL] = baud;
140 #else
141     *serialBaudRegisters[uart] = baud;
142 #endif
143
144     *serialRegisters[uart][SERIALB] = (1 << serialBits[uart][SERIALRXCIE]); // Enable Interrupts
145     *serialRegisters[uart][SERIALB] |= (1 << serialBits[uart][SERIALRXEN]) | (1 << serialBits[uart][
      SERIALTXEN]); // Enable Receiver/Transmitter
146 }
```

4.1.3.7 uint8_t serialRxBufferEmpty (uint8_t *uart*)

Check if the receive buffer is empty.

Parameters

uart	UART Module to check

Returns

1 if buffer is empty, 0 if not.

Definition at line 258 of file serial.c.

```
258                                                 {
259     if (uart >= UART_COUNT)
260         return 0;
261
262     if (rxRead[uart] != rxWrite[uart]) {
263         return 0;
264     } else {
265         return 1;
266     }
267 }
```

4.1.3.8 uint8_t serialRxBufferFull (uint8_t *uart*)

Check if the receive buffer is full.

Parameters

uart	UART Module to check

Returns

1 if buffer is full, 0 if not

Definition at line 251 of file serial.c.

References RX_BUFFER_SIZE.

```
251                                         {
252     if (uart >= UART_COUNT)
253         return 0;
254 
255     return (((rxWrite[uart] + 1) == rxRead[uart]) || ((rxRead[uart] == 0) && ((rxWrite[uart] + 1) ==
      RX_BUFFER_SIZE)));
256 }
```

4.1.3.9 uint8_t serialTxBufferEmpty (uint8_t *uart*)

Check if the transmit buffer is empty.

Parameters

uart	UART Module to check

Returns

1 if buffer is empty, 0 if not.

Definition at line 317 of file serial.c.

Referenced by serialClose().

```
317                                          {
318     if (uart >= UART_COUNT)
319         return 0;
320 
321     if (txRead[uart] != txWrite[uart]) {
322         return 0;
323     } else {
324         return 1;
325     }
326 }
```

4.1.3.10 uint8_t serialTxBufferFull (uint8_t *uart*)

Check if the transmit buffer is full.

Parameters

uart	UART Module to check

Returns

1 if buffer is full, 0 if not

Definition at line 310 of file serial.c.

References TX_BUFFER_SIZE.

Referenced by serialWrite().

```
310                                         {
311     if (uart >= UART_COUNT)
312         return 0;
313
314     return (((txWrite[uart] + 1) == txRead[uart]) || ((txRead[uart] == 0) && ((txWrite[uart] + 1) ==
    TX_BUFFER_SIZE)));
315 }
```

4.1.3.11 void serialWrite (uint8_t *uart*, uint8_t *data*)

Send a byte.

Parameters

uart	UART Module to write to
data	Byte to send

Examples:

test.c.

Definition at line 273 of file serial.c.

References serialTxBufferFull(), and TX_BUFFER_SIZE.

Referenced by serialWriteString().

```
273                                                 {
274     if (uart >= UART_COUNT)
275         return;
276
277 #ifdef SERIALINJECTCR
278     if (data == '\n') {
279         serialWrite(uart, '\r');
280     }
281 #endif
282     while (serialTxBufferFull(uart));
283
284     txBuffer[uart][txWrite[uart]] = data;
285     if (txWrite[uart] < (TX_BUFFER_SIZE - 1)) {
286         txWrite[uart]++;
287     } else {
288         txWrite[uart] = 0;
289     }
290     if (shouldStartTransmission[uart]) {
291         shouldStartTransmission[uart] = 0;
292         *serialRegisters[uart][SERIALB] |= (1 << serialBits[uart][SERIALUDRIE]); // Enable Interrupt
293         *serialRegisters[uart][SERIALA] |= (1 << serialBits[uart][SERIALUDRE]); // Trigger Interrupt
294     }
295 }
```

4.1.3.12 void serialWriteString (uint8_t *uart*, const char * *data*)

Send a string.

Parameters

uart	UART Module to write to
data	Null-Terminated String

Examples:

test.c.

Definition at line 297 of file serial.c.

References serialWrite().

```
297                                                                 {
298     if (uart >= UART_COUNT)
299         return;
300
301     if (data == 0) {
302         serialWriteString(uart, "NULL");
303     } else {
304         while (*data != '\0') {
305             serialWrite(uart, *data++);
306         }
307     }
308 }
```

4.1.3.13 void setFlow (uint8_t *uart*, uint8_t *on*)

Manually change the flow control.

Flow Control has to be compiled into the library!

Parameters

uart	UART Module to operate on
on	1 of on, 0 if off

Definition at line 163 of file serial.c.

References XOFF, and XON.

```
163                                         {
164     if (uart >= UART_COUNT)
165         return;
166
167     if (flow[uart] != on) {
168         if (on == 1) {
169             // Send XON
170             while (sendThisNext[uart] != 0);
171             sendThisNext[uart] = XON;
172             flow[uart] = 1;
173             if (shouldStartTransmission[uart]) {
174                 shouldStartTransmission[uart] = 0;
175                 *serialRegisters[uart][SERIALB] |= (1 << serialBits[uart][SERIALUDRIE]);
176                 *serialRegisters[uart][SERIALA] |= (1 << serialBits[uart][SERIALUDRE]); // Trigger
    Interrupt
177             }
178         } else {
179             // Send XOFF
180             sendThisNext[uart] = XOFF;
181             flow[uart] = 0;
182             if (shouldStartTransmission[uart]) {
183                 shouldStartTransmission[uart] = 0;
184                 *serialRegisters[uart][SERIALB] |= (1 << serialBits[uart][SERIALUDRIE]);
185                 *serialRegisters[uart][SERIALA] |= (1 << serialBits[uart][SERIALUDRE]); // Trigger
    Interrupt
186             }
187         }
188         // Wait till it's transmitted
189         while (*serialRegisters[uart][SERIALB] & (1 << serialBits[uart][SERIALUDRIE]));
190     }
191 }
```

Chapter 5

File Documentation

5.1 serial.c File Reference

UART Library Implementation.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdint.h>
#include "serial.h"
#include "serial_device.h"
```

Macros

- #define RX_BUFFER_SIZE 32

  If you define this, a '\r' (CR) will be put in front of a '\n' (LF) when sending a byte.
- #define TX_BUFFER_SIZE 16

  TX Buffer Size in Bytes (Power of 2)
- #define FLOWCONTROL

  Defining this enables incoming XON XOFF (sends XOFF if rx buff is full)
- #define FLOWMARK 5

  Space remaining to trigger xoff/xon.
- #define XON 0x11

  XON Value.
- #define XOFF 0x13

  XOFF Value.

Functions

- uint8_t serialAvailable (void)

  Get number of available UART modules.
- void serialInit (uint8_t uart, uint16_t baud)

  Initialize the UART Hardware.
- void serialClose (uint8_t uart)

  Stop the UART Hardware.
- void setFlow (uint8_t uart, uint8_t on)

  Manually change the flow control.
- uint8_t serialHasChar (uint8_t uart)

Check if a byte was received.

- uint8_t serialGetBlocking (uint8_t uart)

  Wait until a character is received.

- uint8_t serialGet (uint8_t uart)

  Read a single byte.

- uint8_t serialRxBufferFull (uint8_t uart)

  Check if the receive buffer is full.

- uint8_t serialRxBufferEmpty (uint8_t uart)

  Check if the receive buffer is empty.

- void serialWrite (uint8_t uart, uint8_t data)

  Send a byte.

- void serialWriteString (uint8_t uart, const char *data)

  Send a string.

- uint8_t serialTxBufferFull (uint8_t uart)

  Check if the transmit buffer is full.

- uint8_t serialTxBufferEmpty (uint8_t uart)

  Check if the transmit buffer is empty.

5.1.1 Detailed Description

UART Library Implementation.

Definition in file serial.c.

5.2 serial.h File Reference

UART Library Header File.

Macros

- #define BAUD(baudRate, xtalCpu) ((xtalCpu)/((baudRate)*16l)-1)

  Calculate Baudrate Register Value.

Functions

- uint8_t serialAvailable (void)

  Get number of available UART modules.

- void serialInit (uint8_t uart, uint16_t baud)

  Initialize the UART Hardware.

- void serialClose (uint8_t uart)

  Stop the UART Hardware.

- void setFlow (uint8_t uart, uint8_t on)

  Manually change the flow control.

- uint8_t serialHasChar (uint8_t uart)

  Check if a byte was received.

- uint8_t serialGet (uint8_t uart)

  Read a single byte.

- uint8_t serialGetBlocking (uint8_t uart)

  Wait until a character is received.

- uint8_t serialRxBufferFull (uint8_t uart)

Check if the receive buffer is full.

- uint8_t serialRxBufferEmpty (uint8_t uart)

  Check if the receive buffer is empty.

- void serialWrite (uint8_t uart, uint8_t data)

  Send a byte.

- void serialWriteString (uint8_t uart, const char *data)

  Send a string.

- uint8_t serialTxBufferFull (uint8_t uart)

  Check if the transmit buffer is full.

- uint8_t serialTxBufferEmpty (uint8_t uart)

  Check if the transmit buffer is empty.

5.2.1 Detailed Description

UART Library Header File.

Definition in file serial.h.

5.3 serial_device.h File Reference

UART Library device-specific configuration.

5.3.1 Detailed Description

UART Library device-specific configuration. Contains Register and Bit Positions for different AVR devices.

Definition in file serial_device.h.

Chapter 6

Example Documentation

6.1 test.c

Initializes all available UART Modules.Then prints a welcome message on each and waits for incoming characters, which will be repeated on the UART module they were received on.

```
/*
 * test.c
 *
 * Copyright (c) 2012, Thomas Buck <xythobuz@me.com>
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * - Redistributions of source code must retain the above copyright notice,
 *   this list of conditions and the following disclaimer.
 *
 * - Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
 * TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdint.h>

#include "serial.h"

int main(void) {
    // Initialize UART modules
    for (int i = 0; i < serialAvailable(); i++) {
        serialInit(i, BAUD(38400, F_CPU));
    }

    // Enable Interrupts
    sei();

    // Print Welcome Message
    for (int i = 0; i < serialAvailable(); i++) {
        serialWriteString(i, "Hello from UART");
        serialWrite(i, i + '0');
        serialWriteString(i, "... :)\n");
    }

    // Wait for incoming bytes
    for(;;) {
        for (int i = 0; i < serialAvailable(); i++) {
            if (serialHasChar(i)) {
```

```
                serialWrite(i, serialGet(i));
            }
        }
    }
    return 0;
}
```

Index