Common Use of the AVR Hardware UART

Introduction

This document provides a short introduction to the use of the hardware UART present in most AVR devices. There are several existing application notes regarding the UART (AVR304, AVR305 and AVR306). Issues outside the boundaries of this document might be resolved in one of these.

Overview

The following registers affect the AVR hardware UART:

- **UDR – UART Data Register**
  Actually two physically separated registers sharing the same I/O address. Transmitted and received data are written to, and read from this register.

- **USR – UART Status Register**
  This register contains status information bits, the most commonly used being Receive Complete, Transmit Complete, and Data Register Empty.

- **UCR – UART Control Register**
  In this register, the transmission interrupts are enabled/disabled, as well as the transmitter/receiver themselves. Specifics about the data word are also set here.

- **UBRR – UART Baud Rate Register**
  In this register, the transmission BAUD rate is set. The datasheet on every AVR part contains tables of the most common baudrate settings, as well as general equations to set the correct value in the UBR register if the tables do not cover the parameters.

The operation of the UART is not very complex. In the following, two examples will be presented: Polled and Interrupt controlled UART. The latter can be expanded further as suggested in the application note AVR306.
**Polled UART**

### Code examples

#### C Code

```c
#include definitions for the AT90S8515
#define ENABLE_BIT_DEFINITIONS
#include <io8515.h>

//initialize UART
void InitUART(unsigned char baudrate)
{
    UBRR = baudrate;
    //enable receiver and transmitter
    UCR |= (1<<RXEN)|(1<<TXEN);
}

//receive a byte
unsigned char ReceiveByte(void)
{
    //polls on receive complete
    while(!(USR & (1<<RXC)))
    {
        return UDR;  //wait //return data
    }

    //transmit a byte
    void TransmitByte(unsigned char data)
    {
        //polls on data register empty
        while(!(USR & (1<<UDRE)))
        {
            UDR = data;  //wait //transmit data
        }

        //sample program: echo a character
        void main(void)
        {
            //set the baudrate to 19.200bps@3.686MHz
            InitUART(11);
            while(1)  //eternal loop
            {
                TransmitByte(ReceiveByte());
            }
        }
    }
}
```

#### Assembly

```assembly
;include definitions for the AT90S8515
.include "8515def.inc"

;definitions
.def  temp = r16  ;temporary data

lditemp,low(RAMEND)
outSPL,temp
lditemp,high(RAMEND)
outSPH,temp;init Stack Pointer
rjmpstart;reset handler

;initialize UART
initialize:    ;baudrate in temp
    out UBRR,temp
    ;enable receiver and transmitter
    ldi temp,(1<<RXEN)|(1<<TXEN)
    out UCR,temp
    ret

;receive a byte
receive:
    sbis USR,RXC  ;receive complete?
    rjmp receive
    in temp,UDR ;return data in temp
    ret

;transmit a byte
transmit:
    sbis USR,UDRE ;ready to send?
    rjmp transmit
    out UDR,temp
    ret

;sample program: echo a character
start:
    ldi temp,11
    rcall initialize       ;19.200bps@3.686MHz

loop:
    rcall receive
    rcall transmit
    rjmp loop
```
Interrupt driven UART

**C Code**

```c
#include bit definitions for the AT90S8515
#define ENABLE_BIT_DEFINITIONS
#include <io8515.h>
#include <ina90.h>

//declarations
void TransmitByte(unsigned char data);

//receive complete interrupt
interrupt [UART_RX_vect] void
UART_RX_interrupt(void)
{
    unsigned char data;
    data = UDR;            //receive data
    TransmitByte(data);    //bounce data back
}

//initialize UART
void InitUART(unsigned char baudrate)
{
    UBRR = baudrate;
    /*enable receive complete interrupt,
       receiver and transmitter*/
    UCR |= (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
}

//transmit a byte
void TransmitByte(unsigned char data)
{
    UDR = data;
}

void main(void)
{
    InitUART(11);       //19.200bps@3.686MHz
    while(1)
        ; //eternal loop
}
```

**Assembly**

```assembly
.include 8515def.inc

.def temp = r16        ;temporary data

.org $0000
    lditemp,low(RAMEND)
    outSPL,temp
    lditemp,high(RAMEND)
    outSPH,temp ;init Stack Pointer
    rjmp  start             ;reset handler

.org URXCaddr          ;definition in the
    rjmp UART_RX_interrupt ;8515 include file

;receive complete interrupt
UART_RX_interrupt:
    in    temp ,UDR
    rcall transmit
    reti

;initialize UART
initialize:
    out   UBRR, temp   ;init baudrate
    ;enable receiver, transmitter and TXCint
    ldi   temp, (1<<RXEN)|(1<<TXEN)|(1<<RXCIE)
    out   UCR, temp ;global interrupt enable
    sei
    ret

;transmit a byte
transmit:
    sbis  USR,UDRE ;ready to send?
    rjmp  transmit
    out   UDR, temp
    ret

.start:
    ldi   temp, 11     ;19.200bps@3.686MHz
    rcall initialize

.forever:
    rjmp  forever      ;eternal loop
```