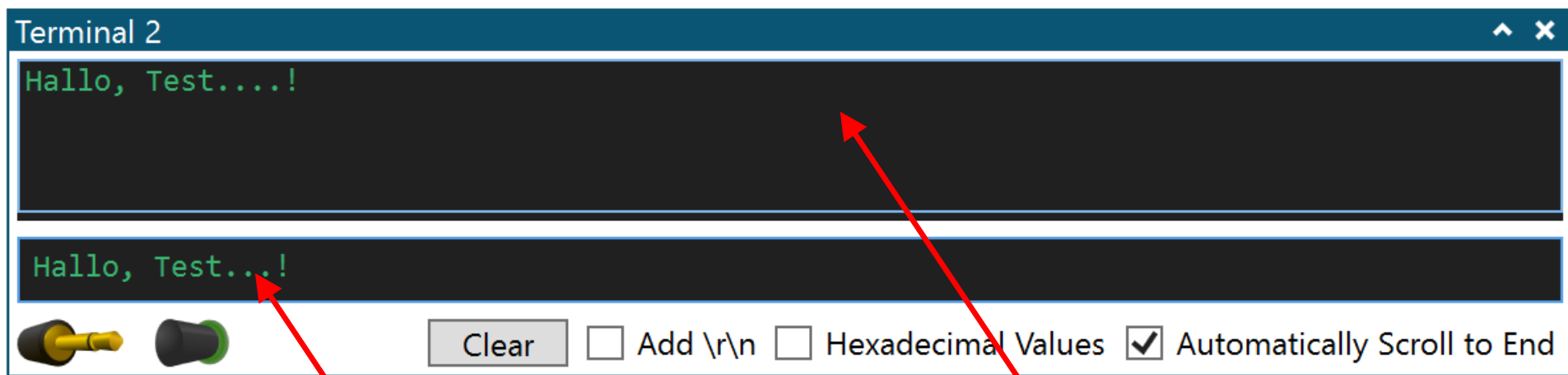


Serielle Schnittstelle, erstes Testprogramm (a)

Verbinden Sie die Mikrocontrollerplatine mit dem USB-Anschluss Ihres Rechners und laden Sie das abgebildete Testprogramm auf den Mikrocontroller.

- Es handelt sich dabei um ein einfaches „Echo-Programm“. Alle über die serielle Schnittstelle an den Mikrocontroller übertragenen Zeichen werden unmittelbar zurück an den PC gesendet.
- Starten Sie ein Terminalprogramm (Atmel Studio → Extras → Data Visualizer, dann auf der linken Seite „External Connection / Serial Port“ wählen).



Eingabe der Daten, die an den Mikrocontroller gesendet werden

Ausgabe der Antwort vom Mikrocontroller

Serielle Schnittstelle, erstes Testprogramm (b)

```
#define F_CPU 16000000UL
#include <avr/io.h>

#define SERIAL_OUT PD1
#define SERIAL_IN PD0

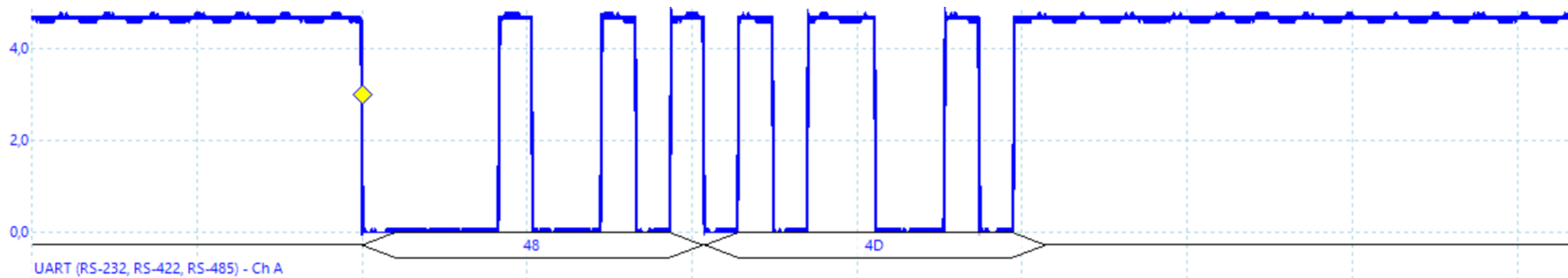
// Serielle Schnittstelle, Echo-Testprogramm
int main(void)
{
    UCSR0B = 0;           // Arduino Nano: Disable Serial Port!!!
    DDRD = 0b0000010;    // Senden/TXD --> PD1, Empfangen/RXD --> PD0

    while(1)
    {
        if(PIND & _BV(SERIAL_IN))
            PORTD |= _BV(SERIAL_OUT);
        else
            PORTD &= ~_BV(SERIAL_OUT);
    }
}
```

Bitmuster für Datenübertragung „manuell“ erzeugen (a)

Erstellen Sie ein neues Projekt/Programm, welches einen Stern („*“) in einer Endlosschleife immer wieder vom Mikrocontroller zum PC überträgt.

- Das Bitmuster für die Datenübertragung soll dadurch erzeugt werden, dass Sie gezielt den Anschluss PD1 (TXD) des Mikrocontrollers auf „1“ oder „0“ setzen. Skizzieren Sie zuvor auf einem Blatt Papier, welche Bits nacheinander gesendet werden müssen und wie lange diese Bits jeweils aktiv sein müssen.
- Die korrekten Zeitintervalle zwischen den einzelnen Bits erzeugen Sie dadurch, dass das Senden der einzelnen Bits innerhalb eines Timer-Interrupts erfolgt. Dazu muss natürlich der Timer zu Beginn des Programms auf eine passende Geschwindigkeit eingestellt werden.
- Wählen Sie als Übertragungsrate 9,6 kbit/s und übertragen Sie die einzelnen Zeichen mit 8 Datenbits, 1 Stoppbit und keinem Paritätsbit.



Bitmuster für Datenübertragung „manuell“ erzeugen (b)

```

#define F_CPU 16000000UL
#include <stdint.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#define SERIAL_OUT PD1 // Anschluss zum Senden
#define SERIAL_IN PD0 // Anschluss zum Empfangen, wird hier nicht benutzt

//          +----- Startbit
//          | +----- Datenbit 0
//          | | +---+---+---+---+---+--- Datenbit 1..6
//          | | | | | | | | +----- Datenbit 7
//          | | | | | | | | | +----- Stoppbit
//          | | | | | | | | | |
int8_t bits_zum_senden[] = { 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
                             0, 1, 0, 1, 1, 0, 0, 1, 0, 1,
                             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, // kleine Pause...
                             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, // kleine Pause...
                             -1 }; // Ende der Übertragung - zurück zum Anfang

// Index des nächsten zu sendenden Bits
int8_t idx = 0;

```

Bitmuster für Datenübertragung „manuell“ erzeugen (c)

```
// Timer Compare Match Interrupt
ISR(TIMER0_COMPA_vect)
{
    if(bits_zum_senden[idx] == 1) PORTD |=  _BV(SERIAL_OUT);
    if(bits_zum_senden[idx] == 0) PORTD &= ~_BV(SERIAL_OUT);
    ++idx;
    if(bits_zum_senden[idx] == -1) idx = 0;
}

// PORTD initialisieren, 9600 Timer-Interrupts pro Sekunde generieren
void init_port_and_timer(void)
{
    UCSR0B = 0;           // Arduino Nano: USART deaktivieren!!!
    DDRD   = _BV(SERIAL_OUT); // TXD-Anschluss als Ausgang aktivieren

    // TODO: 9600 TIMER-INTERRUPTS PRO SEKUNDE GENERIEREN
}

int main(void)
{
    init_port_and_timer();
    while(1) { /* Endlosschleife */ }
}
```

Eingebaute serielle Schnittstelle, USART (a)

Erstellen Sie nochmals ein „Echo-Programm“ für die serielle Schnittstelle. Nun soll das Lesen und Schreiben der Daten mithilfe der im Mikrocontroller eingebauten seriellen Schnittstelle erfolgen (USART, Universal Synchronous Asynchronous Receiver Transceiver)

- Beginnen Sie mit dem vorbereiteten Quelltext auf der folgenden Seite.
- Zur Initialisierung der Schnittstelle, zum Senden von Daten und zum Empfangen von Daten können Sie die im Datenblatt des Mikrocontrollers angegebenen Funktionen verwenden (Datenblatt Kapitel 24.6. bis 24.8., S. 230 ff.)
- Wozu dienen die hier verwendeten Register **UBRR0L**, **UBRR0H**, **UCSR0A**, **UCSR0B**, **UCSR0C** und **UDR0**? Schauen Sie sich zu Hause die betreffenden Abschnitte im Datenblatt des Mikrocontrollers nochmals in Ruhe an.
- **Tipp: Wenn es beim Übersetzen des Programms Fehlermeldungen gibt, kann dies auch an Druckfehlern im Datenblatt des Mikrocontrollers liegen...**

Eingebaute serielle Schnittstelle, USART (b)

```
#define F_CPU 16000000UL
#include <inttypes.h>
#include <avr/io.h>

#define BAUD 9600
#define MYUBRR F_CPU/16/BAUD-1

void USART_Init(uint16_t ubrr)
{
    // USART initialisieren: Datenblatt, 24.6. (S. 230)
}

void USART_Transmit(uint8_t data)
{
    // Ein Zeichen senden: Datenblatt, 24.7. (S. 231)
}

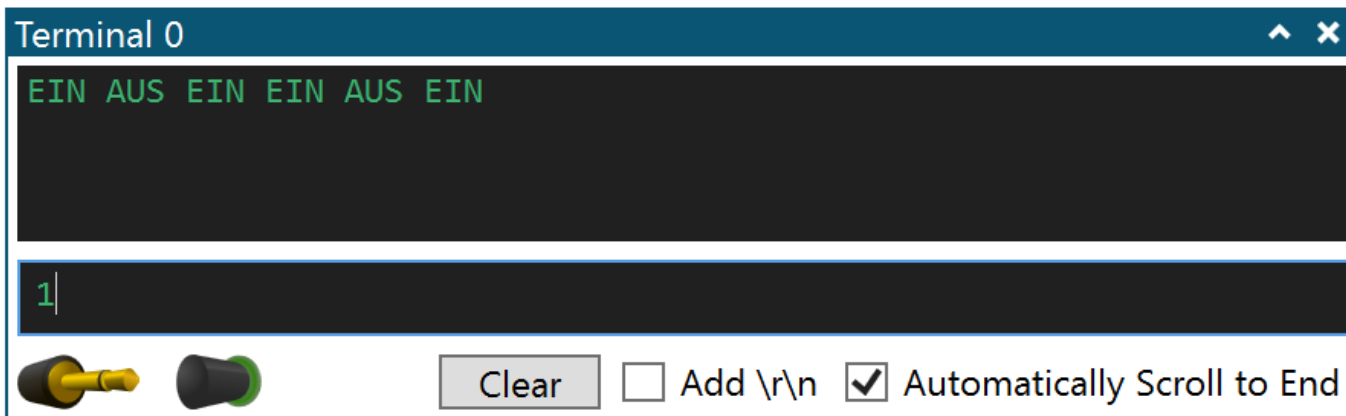
uint8_t USART_Receive(void)
{
    // Ein Zeichen empfangen: Datenblatt, 24.8. (S. 233)
}

int main(void)
{
    USART_Init(MYUBRR);
    while(1) { uint8_t ch = USART_Receive(); USART_Transmit(ch); }
}
```

LED über serielle Schnittstelle ein-/ausschalten

Erstellen Sie ein Projekt/Programm zum Ein- bzw. Ausschalten einer Leuchtdiode über die serielle Schnittstelle.

- Die Leuchtdiode soll eingeschaltet werden, falls der Anwender die Ziffer „1“ über die Schnittstelle sendet.
- Die Leuchtdiode soll wieder ausgeschaltet werden, falls der Anwender die Ziffer „0“ über die Schnittstelle sendet.
- Sie entscheiden, ob Sie „nur“ die interne Leuchtdiode am Anschluss PB5 oder auch eine externe Leuchtdiode (bitte mit Vorwiderstand!) ansteuern möchten.
- Wenn die Leuchtdiode eingeschaltet wird, sendet der Mikrocontroller den Text „EIN“ über die Schnittstelle zurück an den PC. Wenn die Leuchtdiode ausgeschaltet wird, sendet der Mikrocontroller den Text „AUS“.

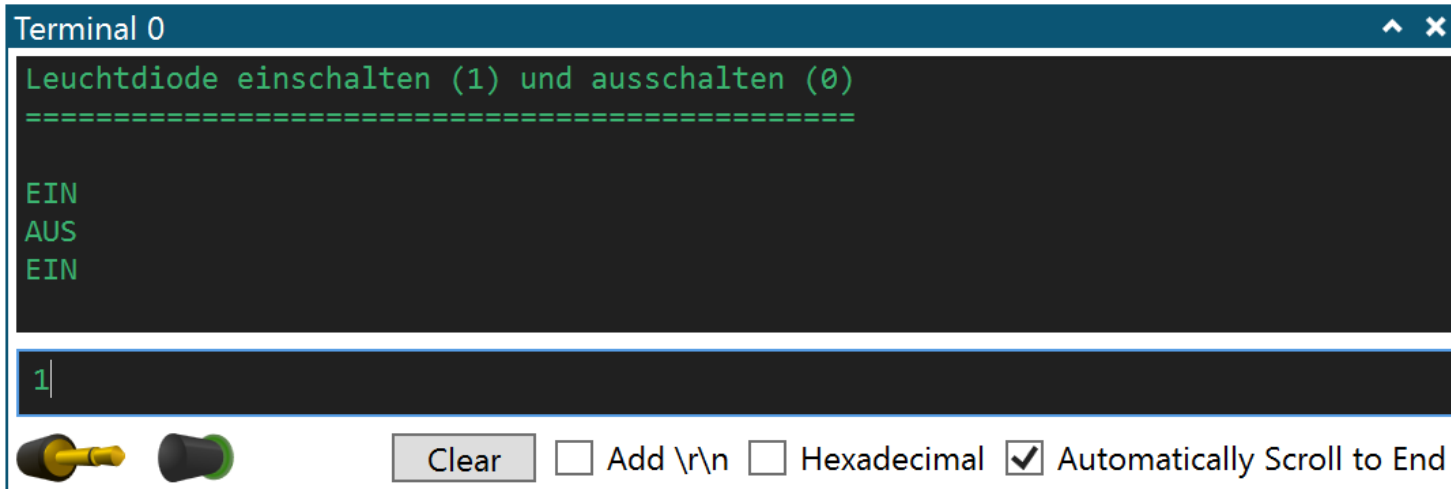


```
Terminal 0
EIN AUS EIN EIN AUS EIN
1
Clear  Add \r\n  Automatically Scroll to End
```


Senden von Zeichenketten, Zusatzaufgabe

Schreiben Sie eine zusätzliche Funktion `USART_TransmitLine(text)` zum Senden von kompletten Zeichenketten. Nach der Übertragung einer Zeichenkette soll automatisch ein Zeilenumbruch gesendet werden.

- Erweitern Sie das Programm zum Ein-/Ausschalten einer LED, sodass zu Beginn eine Begrüßungsmeldung ausgegeben wird. Die Statusmeldungen („EIN“ bzw. „AUS“) sollen nun untereinander ausgegeben werden.
- Tipp: Das Ende der Zeichenkette erkennen Sie am Nullbyte.



```
Terminal 0
Leuchtdiode einschalten (1) und ausschalten (0)
=====
EIN
AUS
EIN
1
```

```
void USART_TransmitLine(const char *text)
{
    // TODO: Komplette Textzeile inkl. Zeilenumbruch senden...
}
```