Informatikwerkstatt, Foliensatz 2
Serieller Datenaustausch

G. Kemnitz

Institut für Informatik, Technische Universität Clausthal

13. November 2013

Inhalt des Foliensatzes

LCD-Ausgabe

PC-Verbindung über USART/USB-Converter

Drahtlose Kommunikation über Bluetooth

Zielstellung

Das Ausprobieren von Programmteilen, Kontrollausgaben und die Steuerung des zu entwickelnden Fahrzeugs erfordert die Ein- und Ausgabe von Zahlen und Texten.

Zahlenformate	1 Byte	2 Byte	4 Byte
ohne Vorzeichen	0 bis 255	0 bis $2^{16}-1$	0 bis $2^{32}-1$
mit Vorzeichen	-128 bis 127	-2^{15} bis $2^{15}-1$	-2^{31} bis $2^{31}-1$

Textdarstellung durch eine Bytefolge (ASCII):

Zeichen	H	a	l	l	o	!
Zahlenwert	72	97	108	108	111	33
hexadezimal	0x48	0x61	0x6C	0x6C	0x6F	0x20

Kontrollfragen

- Wie erfolgt die Umrechnung zwischen dezimal und hexadezimal?
- Wie lautet die Hexadezimal- und die Binärdarstellung der folgenden Dezimalzahlen?

  `1023, 17, 56`
- Welche Dezimalwert haben die folgenden Hexadezimalzahlen?

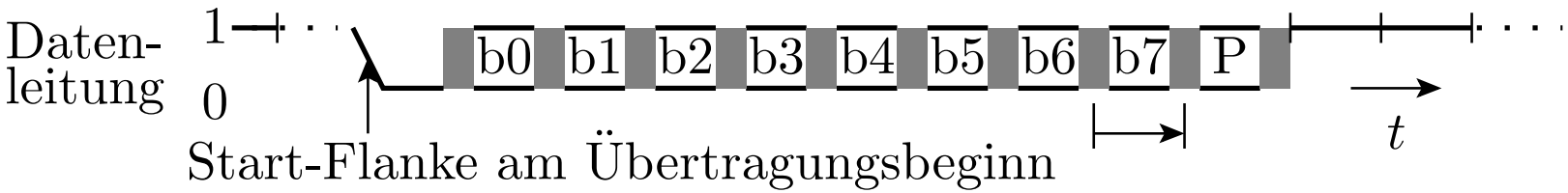
  `0x20, 0xFF, 0x18`
- Welche Zeichenfolge stellt die folgende Zahlenfolge dar[1]?

  `0x54, 0x6E, 0x63, 0x68, 0x6E, 0x69, 0x78`

[1] 'A' hat den Wert 0x41 und 'a' den Wert 0x61. Weitere Wertezuordnung in der Reihenfolge der Buchstaben.

Serieller Datenaustausch

Rechner und Rechnerbausteine tauschen ihre Daten oft seriell[2] aus. Ein vielgenutztes Protokoll[3] für die byteweise Übertragung:

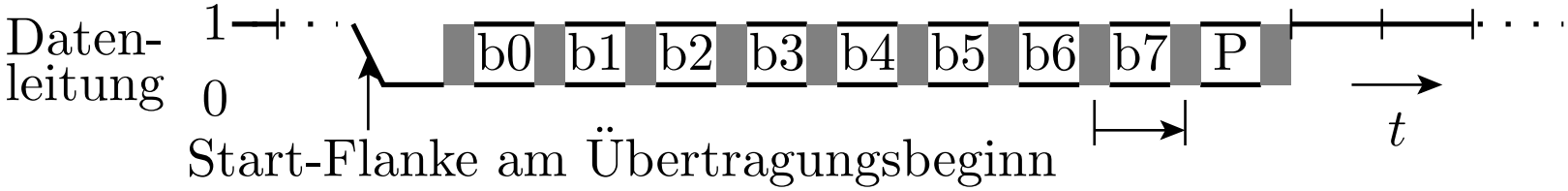


$b \in \{0,1\}$	Datenbits	$\cdots$	Übertragungspause
$P \in \{0,1\}$	Paritätsbit	$\longmapsto$	Bitzeit, z.B. $t_{\text{Bit}} \approx 0{,}1\,\text{ms}$
$\vdash\!\!\dashv$	Stoppbit (Wert 1)		Übertragungsdauer: $12 \cdot t_{\text{Bit}}$

Der Empfänger erkennt den Übertagungsbeginn an der Stopp/Start-Flanke und übernimmt die Werte nach 1,5, 2,5 etc. Bitzeiten. Voraussetzung: Gleich eingestellte Bitzeit, Bitanzahl, Stoppbitanzahl und Parität bei Sender und Empfänger.

[2] Seriell: Hintereinander über eine, statt parallel über viele Leitungen.

[3] Kommunikationsprotokoll: Vereinbarung, nach der die Datenübertragung zwischen zwei oder mehr Teilnehmern erfolgt.



$b \in \{0,1\}$	Datenbits	$\cdots$	Übertragungspause
$P \in \{0,1\}$	Paritätsbit	$\longmapsto$	Bitzeit, z.B. $t_{\text{Bit}} \approx 0{,}1\,\text{ms}$
$\dashv\vdash$	Stoppbit (Wert 1)		Übertragungsdauer: $12 \cdot t_{\text{Bit}}$

Erkennbare Protokollfehler:

- Framefehler: Startbit nicht lange genug null. Stoppbits nicht lange genug eins. Änderung der Bitwerte im Abtastintervall.
- Paritätsfehler: bei (un)gerader Parität, Anzahl der empfangenen Einsen nicht (un)gerade.
- Datenüberlauf: Empfangendes Byte wird vor dem Abholen von Bits des nächsten Bytes überschrieben.

- Protokollfehler verursachen oft Empfangsfehler.
- Protokollfehlerbehandlung ist extra zuprogrammieren.

Serielle Schnittstellen des ATmega64

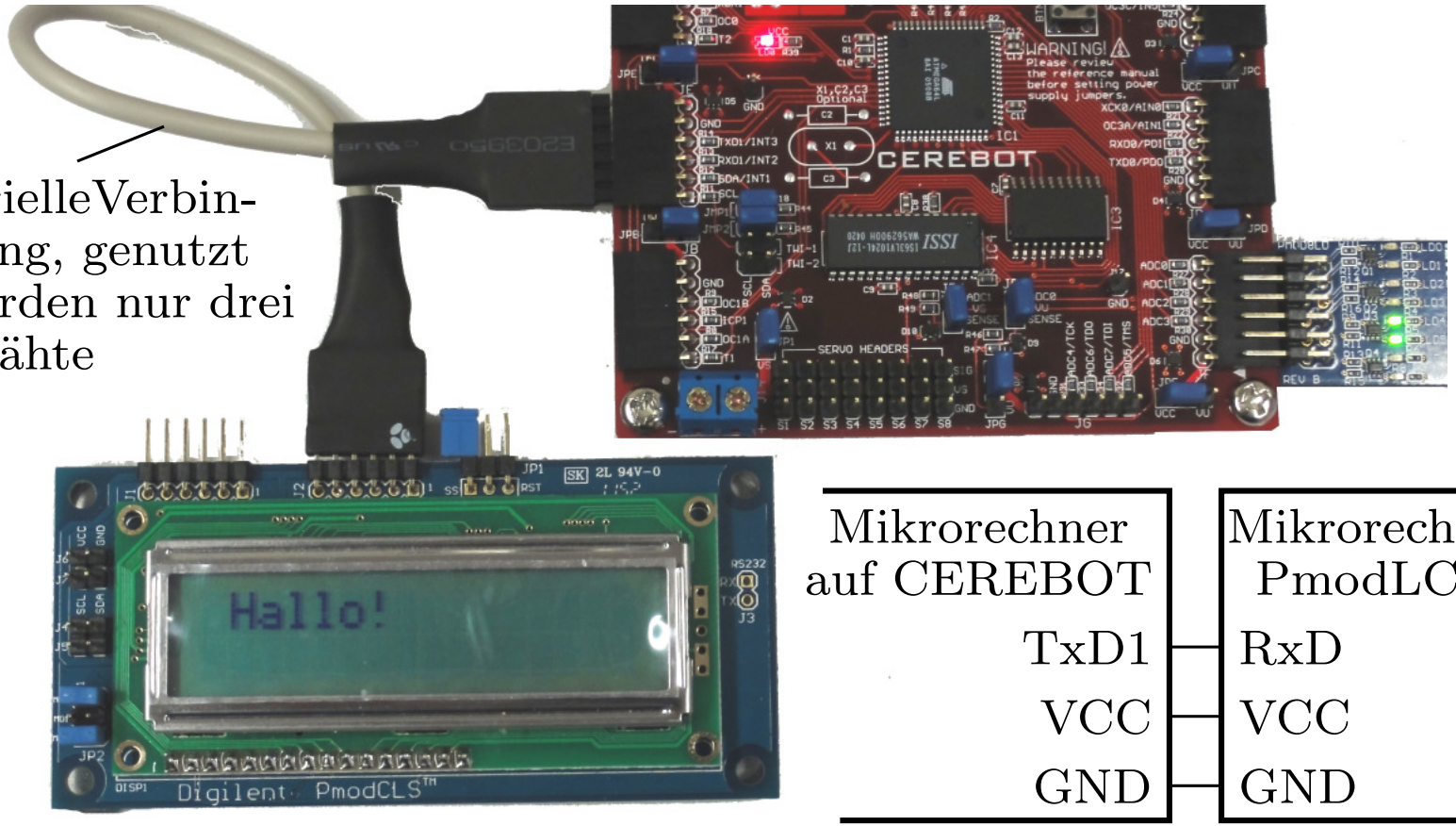
- 2x USART (Universal Synchronous and Asynchronous Receiver and Transmitter)
- 1x SPI (Serial Peripherial Interface)

Vorgesehener Einsatz in den geplanten Projekten:

- USARTs mit dem beschriebenen asynchronen Protkoll:
  - LC-Display-Ausgabe
  - Anschluss über einen Seriell/USB-Konverter an den PC
  - drahtlose Kommunikation mit PC über Bluetooth
- SPI-Schnittstelle (mit einem synchronen Protokoll)
  - Anschluss eines Joysticks.

LCD-Ausgabe

Anschluss des LC-Displays



Mikrorechner auf CEREBOT		Mikrorechner PmodLCD
TxD1	—	RxD
VCC	—	VCC
GND	—	GND

TxD1	Sendesignal USART1	VCC	Versorgungsspannung
RxD	Empfangssignal LCD	GND	Masse (Ground)

Konfiguration von USART1 auf CEREBOT

Mit den Jumberstellungen wie auf dem Bild ist im Mikrokontoller des LCD-Moduls als Protokoll eingestellt: asynchron, 9600 Baud, 8 Datenbits, 1 Stoppbit, kein Paritätsbit.

USART1 des ATmega64 ist genauso zu konfigurieren:

- Einstellen des Teilerwerts für die Baudrate

$$T = \frac{f_{\text{Proz}}}{16 \cdot b \cdot \text{Hz}} - 1 = \frac{\approx 7{,}5\,\text{MHz}}{16 \cdot 9600\,\text{Hz}} - 1 = 49$$

(f_{Proz} – Taktfrequenz des Prozessors; b – Baudrate)

```
UBRR1H = 0; UBRR1L = 49;
```

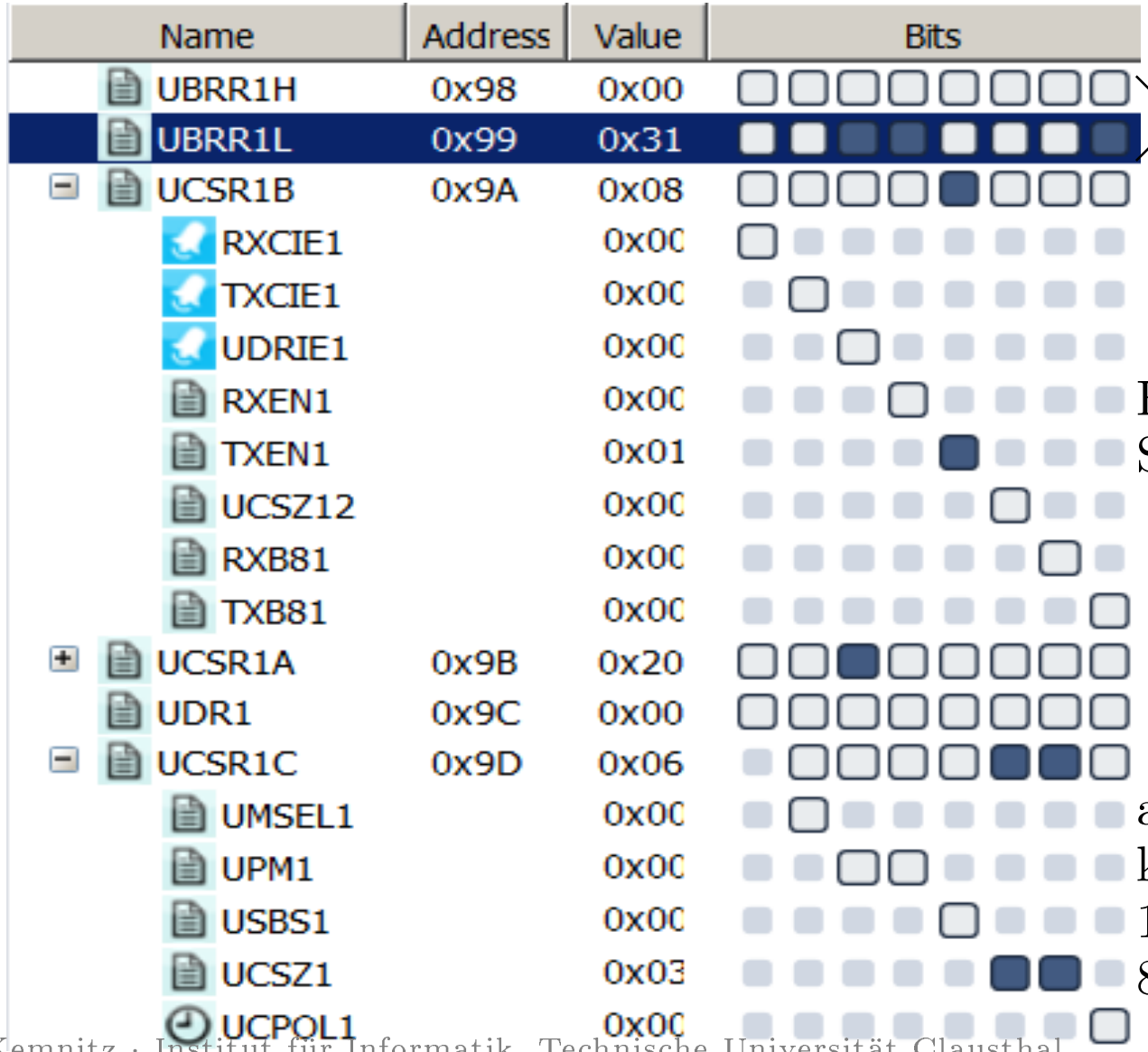
- im Register UCSR1B Sender einschalten:

```
UCSR1B = (1<<TXEN1); // TXEN1 in UCSR1B setzen
```

- asynchron, 8 Daten-, 1 Stoppbit, keine Parität einstellen:

```
UCSR1A = 0b00000110;
```

Die Konfigurationseinstellungen lassen sich in der Debug-IO-View gut kontrollieren und auch korrigieren:



Baudrate 9600

Empfang aktivieren[1]

Senden aktiviert

asynchron Übertragun

kein Paritätsbit

1 Stoppbit

8 Datenbits

[1] für bidirektionale Verbindungen

Programm zur LCD-Ausgabe des Textes »Hallo!«

Ein Text ist in C ein Feld von Byte-Werten, das mit null abgeschlossen ist.

Adresse	Wert	Bedeutung
Text	0x1B	Anzeige löschen
Text+1	0x5B	
Text+2	0x6A	
Text+3	0x48	H
Text+4	0x61	a
Text+5	0x6C	l
Text+6	0xrC	l
Text+7	0x6F	o
Text+8	0x21	!
Text+9	0x00	Ende

```
#include ...//Header einbinden
int main(){
// Vereinbahrung lokaler Variablen
// * Zeichenkette mit Ausgabetext
// * Zeiger auf aktuelles Zeichen
 uint8_t Text[] = "\x1b[i Hallo!";
 uint8_t *prt = Text;
// wiederhole bis Textende
 while (...) {
    ... // warte bis Sendepuffer frei
    ... // Sende aktuelles Zeichen
    ... // Zeiger weiterschalten
 }      // Schleifenende
}       // Programmende
```

Sceenshoot Programms und Anfangswerte der lokalen Variablen.

```
1   #include <avr/io.h>
2   int main(void)  {
3     // Protokolleinstellung USART1 wie beschrieben
4     UBRR1H = 0; UBRR1L = 49;
5     UCSR1B = (1<<TXEN1);
6     UCSR1C = 0b00000110;
7
8     // lokale Variablen wie beschrieben
9     uint8_t Text[] = "\x1b[j Hallo!";
10    uint8_t *ptr = Text;
11
12    // Schleife zum Versenden aller Zeichen
13    while( *ptr != 0) {
14      // warten bis der Puffer leer
15      // (das Bit UDRE1 gesetzt) ist
16      while (!(UCSR1A & (1<<UDRE1))){}
17
18      // adressierte Byte versenden
19      UDR1 = *ptr;
20      ptr++; // Zeiger weiterschalten
21    }
22  }
```

Locals

Name	Value
Text	… @0x10f
[0]	0x1b
[1]	0x5b
[2]	0x6a
[3]	0x20
[4]	0x48
[5]	0x61
[6]	0x6c
[7]	0x6c
[8]	0x6f
[9]	0x21
[10]	0x00
ptr	0x10f1

Aufgabe 2.1: Test der LCD-Ausgabe

- Neues Projekt »LCD« anlegen. Programm von der Folie zuvor eingeben. Zeilennummern einschalten:

  `Tools > Options > Text Editor > All languages > General: Display on line numbers`√

- Unterbrechungspunkte wie im Bild setzen.
- Start Debug bis zum ersten Unterbrechungpunkt, IO-View USART-Einstellungen kontrollieren.
- Einzelschritt, Locals (lokale Variablen) Test und Zeigerwert kontrollieren.
- Programmfortsetzung bis Unterbrechungspunkt; in Local Zeiger und auf dem LCD Ausgabe kontrollieren.
- Test wiederholen und schauen, nach welchem übertragenen Zeichen das LCD gelöscht wird.
- Ausgabe modifizieren.

Aufgabe 2.2: Schaltbare Ausgabe (Fortgeschrittene)

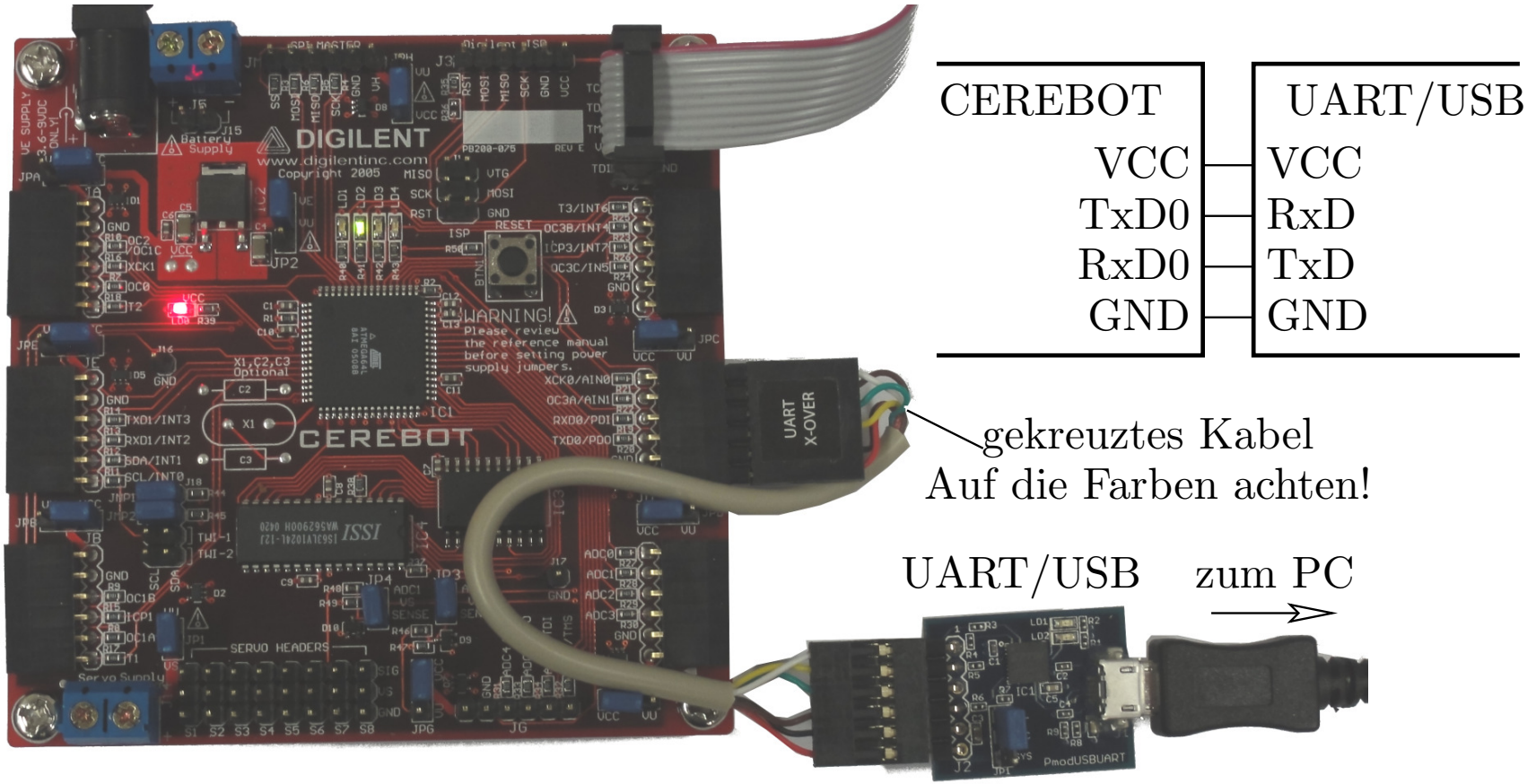
- Stecken Sie zusätzlich das Schaltermodul an Port F.
- Schreiben Sie ein Programm, das bei unterschiedlichen Schalterstellungen unterschiedliche Texte ausgibt.

Hinweis: Fallunterscheidung mit Switch-Anweisung

```
uint8_t sw; DDRF = 0; // alle Bits Eingänge
...
sw = PINF & 0xF; // vier Schalterwerte einlesen
switch (sw){
 case 0b00000001: // SW1 ein, der Rest aus
   <Anweisungen für Fall 1>
   break;
 case 0b00000010: // SW2 ein, der Rest aus
   <Anweisungen für Fall 2>
   break;
...
}
```

PC-Verbindung über USART/USB-Converter

Anschluss des PCs über USART-USB-Konverter



Anschluss des USART/USB-Konverters mit beiliegendem X-Over-Kabel an Stecker JD (USART0).

Konfiguration UART0

- UBRR0H und UBRR0L: Baudrate 9600 einstellen
- UCSR0A: Sender und Empfänger eingeschalten.
- UCSR0C: 8 Daten-, 2 Stoppbits und ungerade Parität

Name	Address	Value	Bits	
UBRR0L	0x29	0x31		Baudrate 9600
⊞ UCSR0B	0x2A	0x18		Send., Empf. ein
⊞ UCSR0A	0x2B	0x20		
UDR0	0x2C	0x00		
UBRR0H	0x90	0x00		Baudrate 9600
⊟ UCSR0C	0x95	0x3E		
UMSEL0		0x0C		asynchron
UPM0		0x03		ungerade Parität
USBS0		0x01		2 Stoppbit
UCSZ0		0x03		8 Datenbit
UCPOL0		0x0C		

Echo-Programm

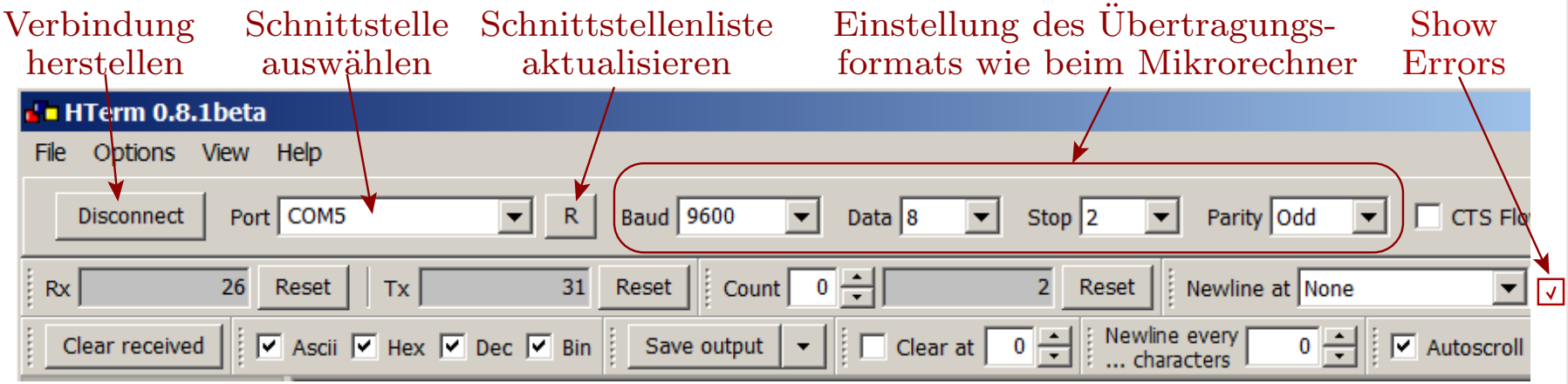
Der Mikrocontroller soll ständig auf Datenbytes warten, deren Wert + 1 und zurücksendet und ... (Was passiert noch?)

```
#include <avr/io.h>
int main(void)  {
  UBRR0H = 0; UBRR0L = 49;
  UCSR0B = 0b00011000;
  UCSR0C = 0b00111110;
  DDRE = 0xF0;

  uint8_t daten, Ct=0;
  while(1) {
    while (!(UCSR0A & (1<<RXC0))){} // warte auf Receive-Bit
    daten = UDR0;                   // empfangene Daten lesen
    while (!(UCSR0A & (1<<UDRE0))){}// warten bis Sendepuffer frei
    UDR0 = daten+1;                 // Daten+1 senden
    PORTE = Ct<<4; Ct++;            // Zähler ausgeben und erhöhen
  }
}
```

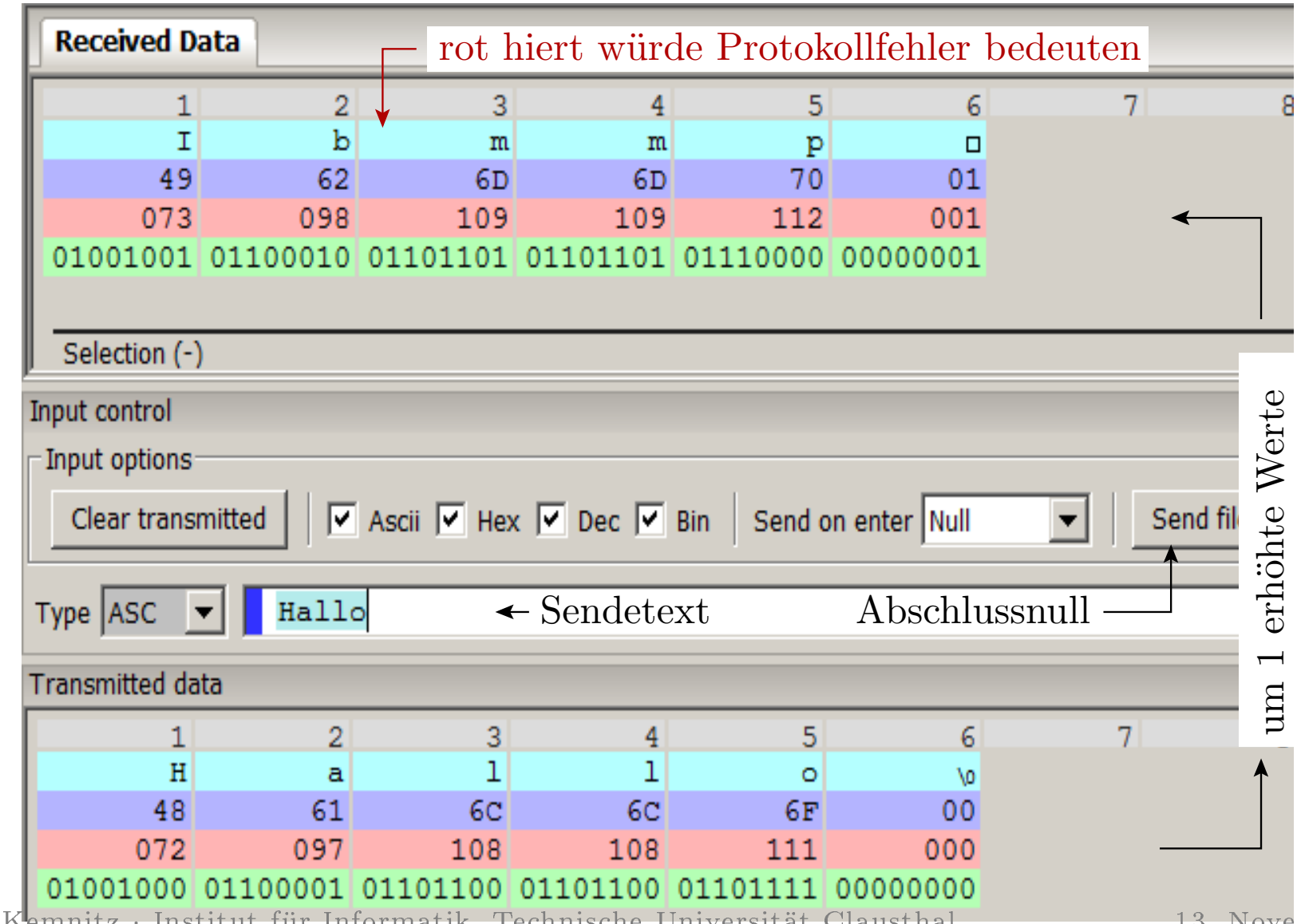
Gegenstelle auf dem PC vorbereiten

HTerm starten. Übertragungsparameter einstellen.

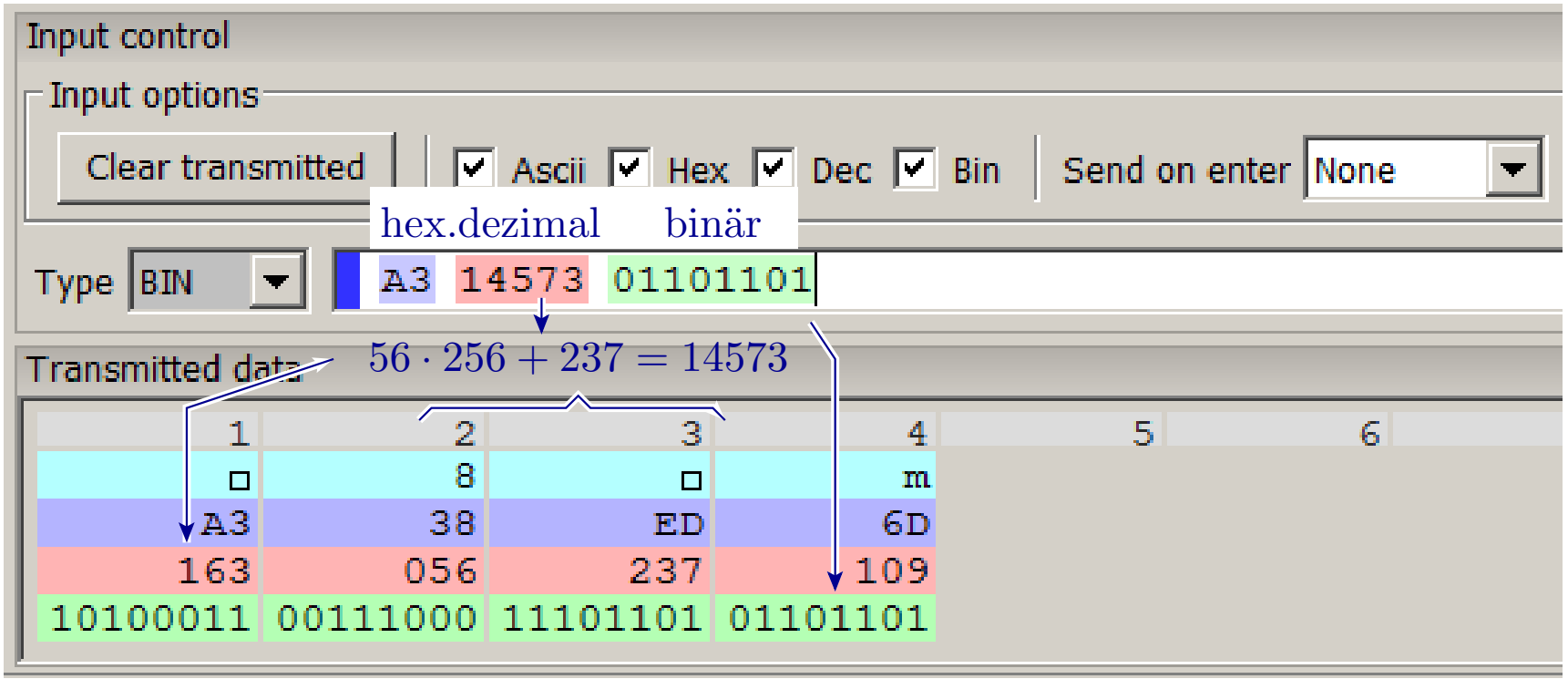


- Der Port kann variieren. Man erkennt ihn an der Änderung der Port-Liste, wenn der USB-Stecker angesteckt wird.
- Show Errors (Empfangsfehleranzeige) aktivieren. Empfangende Daten mit Protokollfehlern werden dann rot dargestellt.
- Bei Protokollfehlern Einstellungen überprüfen, Mikrorechnerprogramm Neustart, ...

Schicken eines Textes



Sendedaten mit gemischter Zahlendarstellung



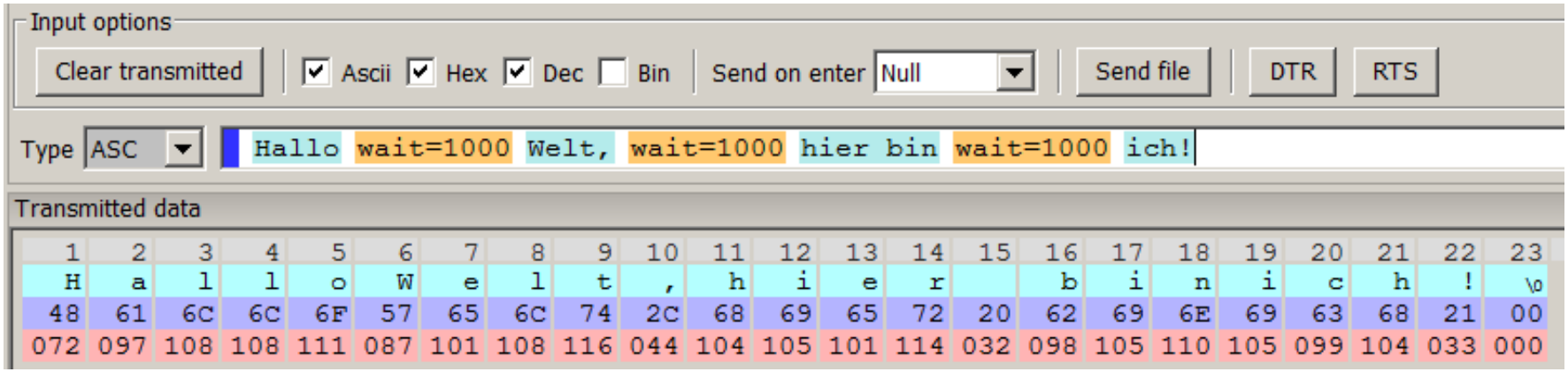
- Dezimalzahlen größer 255 werden als 2-Byte-Zahl übertragen.

Wartezeiten zwischen gesendeten Daten

Bei Mikrorechneranwendungen sind Zeitabläufe wichtig.
Möglichkeit:

- Wartezeiten zwischen Eingabe-Bytes einzufügen und
- die Ausgabe mit Zeitstempel in eine Datei zu speichern, z.B. zur graphischen Darstellung von Sensorsignalverläufen oder der Fahrzeugbewegung.

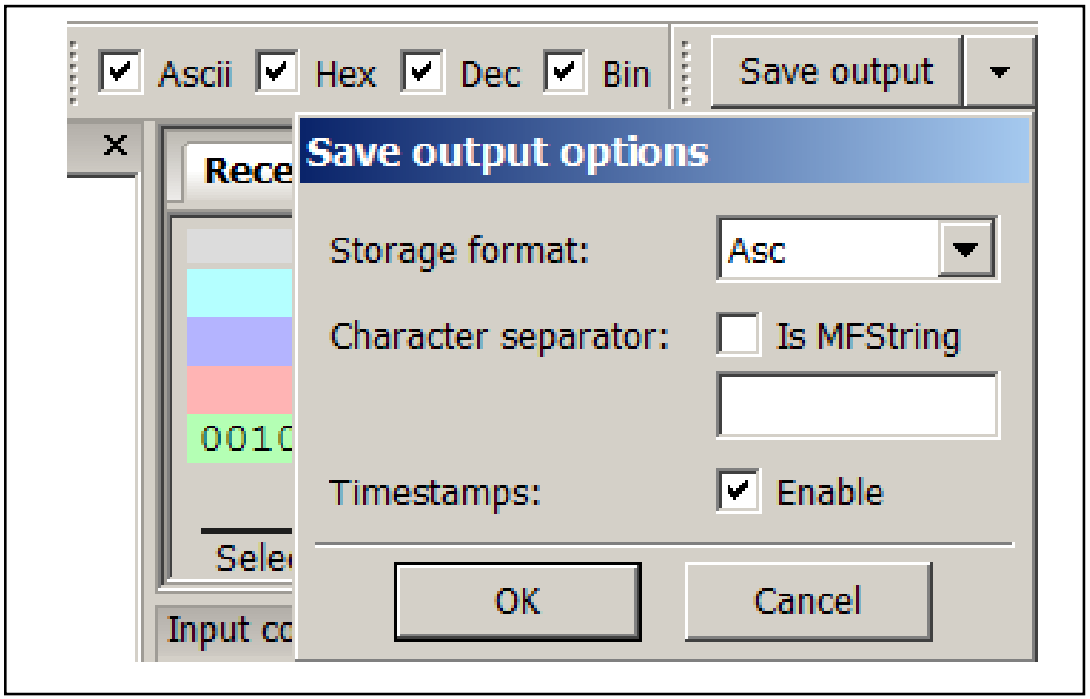
Bytefolge mit 1000 ms Wartezeiten zwischen Bytegruppen:



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
H	a	l	l	o	W	e	l	t	,	h	i	e	r		b	i	n	i	c	h	!	\0
48	61	6C	6C	6F	57	65	6C	74	2C	68	69	65	72	20	62	69	6E	69	63	68	21	00
072	097	108	108	111	087	101	108	116	044	104	105	101	114	032	098	105	110	105	099	104	033	000

Die empfangenen Daten mit Zeitstempel in Datei speichern[4].

Ausgabeformat: Ascii mit Zeitstempel



19:30:06.004: Empfangszeit (h:min:s:ms)
Hal empfangende Zeichenfolge

Ausgabefile

```
Hal
19:30:06.020:
lo
19:30:07.009:
Welt,
19:30:08.015:
hier bin
19:30:09.004:
ich
19:30:09.020:
!
```

[4]Im Beispiel wurde die +1 Erhöhung beim Rücksenden aus dem Mikrorechnerprogramm entfernt.

Aufgabe 2.3: Test der PC-Kommunikation

Testen Sie die besprochenen Beispiele:

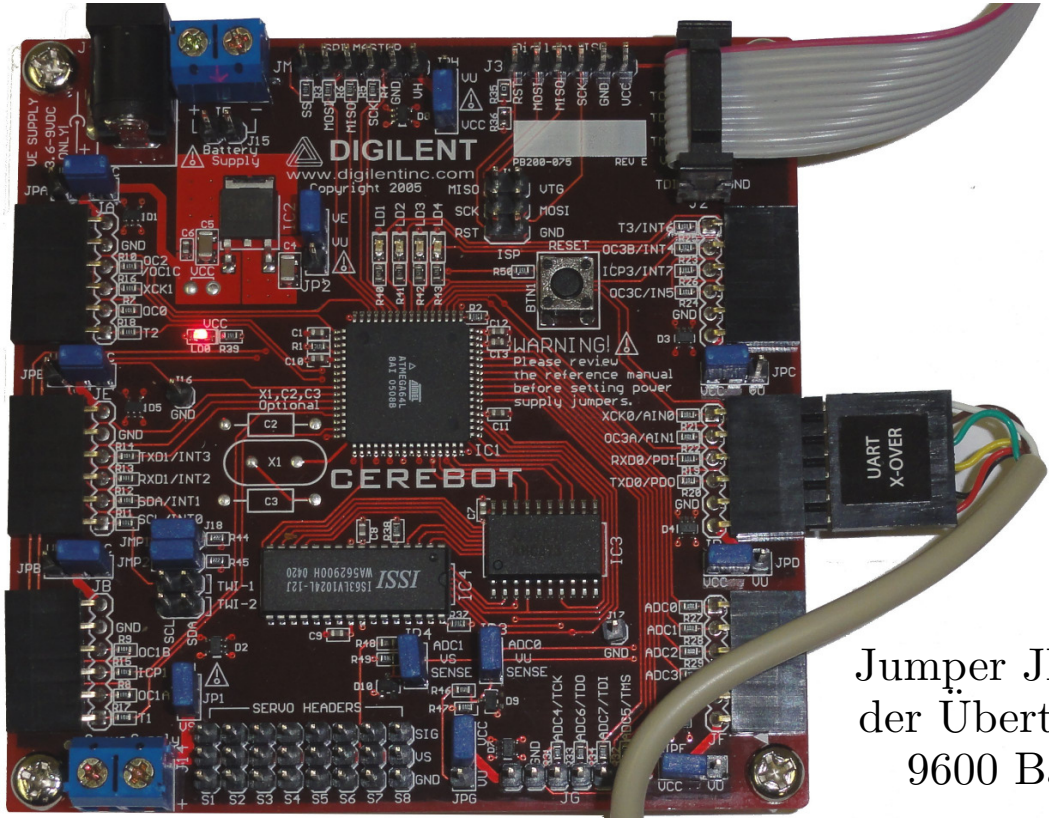
- Anschluss des PModUSBUART und des PCs wie auf Folie 17
- Anlegen eines neuene Projekts »Echo«, Eingabe und Start des Programms auf Folie 19
- Start und Konfiguration des HTerms wie auf Folie 20
- Test mit einer Beispielzeichenkette wie auf Folie 21
- Test mit einer Zahlenfolge wie auf Folie 22.
- Test mit Warteanweisungen wie auf Folie 23 und Dateiausgabe mit Zeitstempel.
- Änderungen der Übertragungsprameter und Tests wiederholen. Empfangsfehler beobachten.

Aufgabe 2.4: PC-LCD-Ausgabe

- Stecken Sie zusätzlich das Modul mit dem LC-Display an USART1 der Mikrorechnerbaugruppe (siehe Folie 9).
- Schreiben Sie ein Programm, dass den Eingabetext vom PC auf das LC-Display ausgibt. In der ersten Version sollen die Steuerzeichen für das LCD. (Bildschirmlöschen etc. vom PC gesendet werden.)
- Suchen Sie im Internet das Referenzmanual des LCD-Moduls (PmodCLS_rm.pdf) und testen Sie einige der Steuerbefehle (Bildschirm löschen, Cursor bewegen, Scollen, Hintergrundbeleuchtung ein-/ausschalten etc.). Bitte ohne vorherige Rücksprache mit dem Übungsleiter keine Änderungen im EEPROM vornehmen.
- Schreiben Sie ein kreatives Programm, in dem Sie das Gelernte über die PC-Kommunikation, die LCD-Ausgabe und Steuerbefehle für das LC-Display anwenden.

Drahtlose Kommunikation über Bluetooth

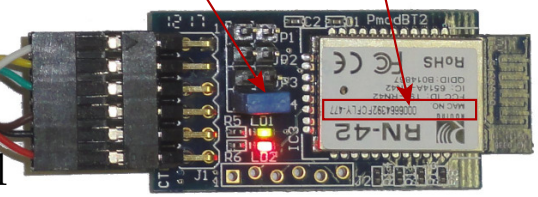
Bluetooth-Modul anschließen



Jumper JP4 zur Einstellung der Übertragungsparameter 9600 Baud 8n1 stecken

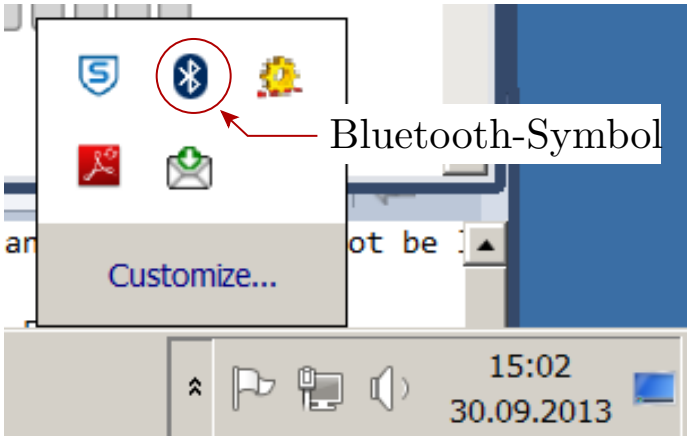
MAC-Nummer

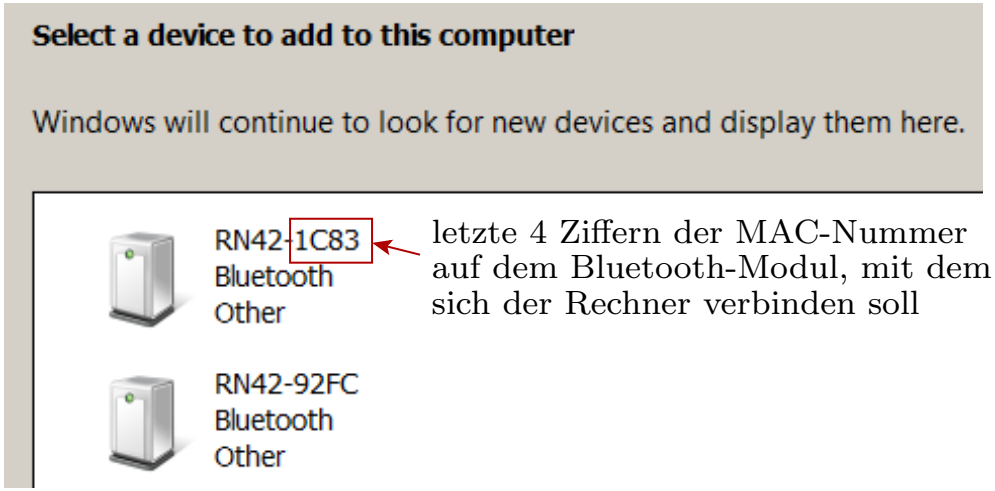
USB-Bluetooth-Dongle



Bluetooth-Modul

Bluetooth-Verbindung auf PC einrichten





- Unter Windows auf Doppelklick auf Bluetooth-Symbol
- bei Bedarf »add Device«
- Device mit der MAC-Nummer auf dem PMOD auswählen
- Bei »Enter the Device Pairing Code« Eingabe »1234«[5]
- recher Mouseklick > Properties > Hardware > hinterm Namen COM-Port ablesen.

Protokoll der seriellen Schnittstelle der Bluetooth-Module: 8 Datenbits, 1 Stoppbit, keine Parität, 9600Baud.

Zum Test der Kommunikation ist der USART/USB-Converter durch das Bluetooth-Modul zu ersetzen und im bisher genutzten Echo-Programm die Stoppbitanzahl auf 1 und die Parität auf keine zu ändern.

```
#include <avr/io.h>
int main(void)  {
  UBRR0H = 0; UBRR0L = 49;
  UCSR0B = 0b00011000;
  UCSR0C = 0b00111110;
  DDRE = 0xF0;

  uint8_t daten, Ct=0;
  while(1) {
    while (!(UCSR0A & (1<<RXC0))){} // warte auf Receive-Bit
    daten = UDR0;                   // empfangene Daten lesen
    while (!(UCSR0A & (1<<UDRE0))){}// warten bis Sendepuffer frei
    UDR0 = daten+1;                 // Daten+1 senden
    PORTE = Ct<<4; Ct++;            // Zähler ausgeben und erhöhen
  }
}
```

0 (nur ein Stoppbit)

00 (keine Parität)

Aufgabe 2.5: Bluetooth-Test

- Stecken Sie das Bluetooth-Modul wie auf Folie 28 an USART 0 (Stecker JD) und den USB-Bluetooth-Dongle in den PC.
- Stellen Sie in der beschriebenen Weise eine Bluetooth-Verbindung her.
- Ändern Sie die Protokollparameter im PC-Programm und im HTerm auf 8 Datenbits, 1 Stoppbit, keine Parität, 9600Baud und öffnen Sie die Verbindung mit dem COM-Port der Bluetooth-Verbindung.
- Führen Sie änliche Tests wie bei der drahtgebundenen Kommunikation durch.