



Informatikwerkstatt, Foliensatz 2

Serieller Datenaustausch

G. Kemnitzv

Institut für Informatik, Technische Universität Clausthal
28. Oktober 2014



Inhalt des Foliensatzes

LCD-Ausgabe

PC-Verbindung über USART/USB-Converter

Drahtlose Kommunikation über Bluetooth

Zahlen- und Textdarstellung

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

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



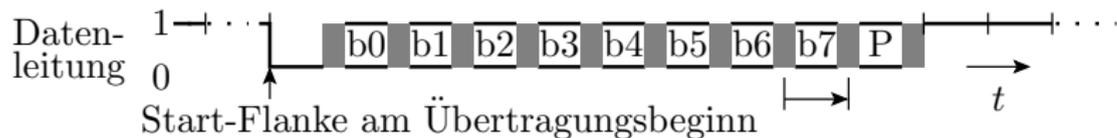
Kontrollfragen

- Wie erfolgt die Umrechnung zwischen Dezimal- und Hexadezimalzahlen?
- Wie lautet die Hexadezimal- und die Binärdarstellung der folgenden Dezimalzahlen?
1023, 17, 56
- Welche Dezimalwert haben die folgenden Hexadezimalzahlen (vorzeichenfreier Zahlentyp)?
0x20, 0xFF, 0x18
- Welche Zeichenfolge stellt die folgende Zahlenfolge dar¹?
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 seriell² aus. Ein vielgenutztes Protokoll³ für die byteweise Übertragung:

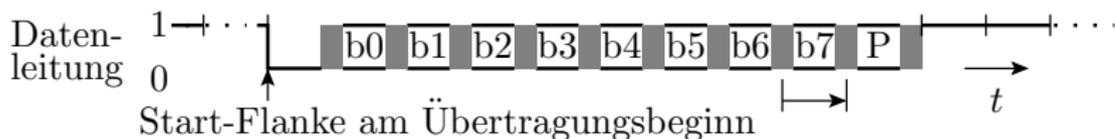


$b \in \{0, 1\}$	Datenbits	\dots	Übertragungspause
$P \in \{0, 1\}$	Paritätsbit	\longrightarrow	Bitzeit, z.B. $t_{\text{Bit}} \approx 0,1 \text{ ms}$
\longleftarrow	Stoppbit (Wert 1)		Übertragungsdauer: $12 \cdot t_{\text{Bit}}$

Der Empfänger erkennt den Übertragungsbeginn an der 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.

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

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



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

Erkennbare Protokollfehler:

- Frame-Fehler: Dauer des Startbits oder der Stoppbits zu kurz. 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 zu programmieren.



Serielle Schnittstellen des ATmega64

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

Vorgesehener Einsatz in den geplanten Projekten:

- USARTs mit dem beschriebenen asynchronen Protokoll:
 - 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.



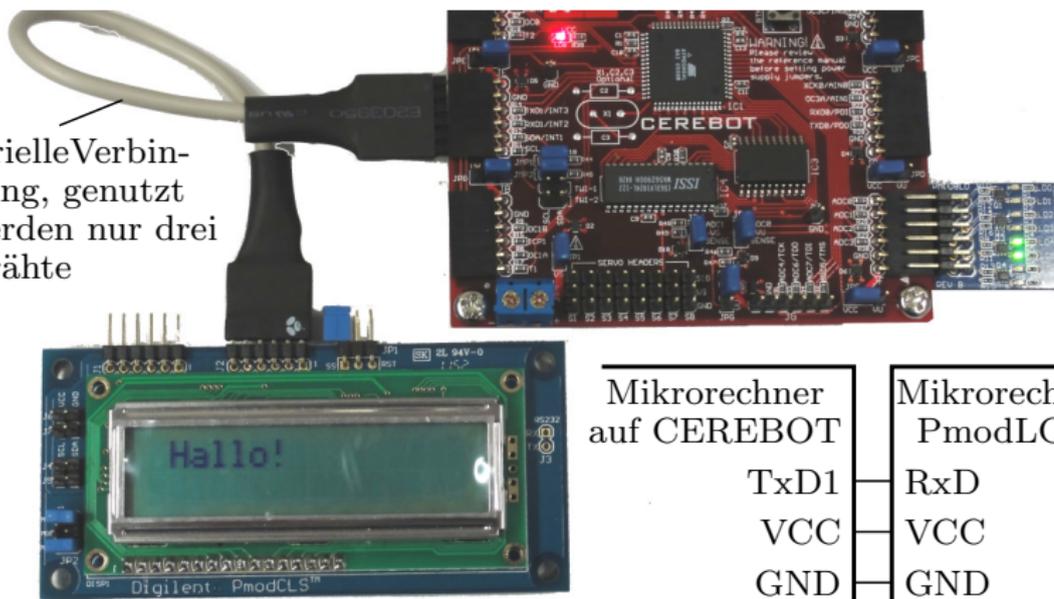
LCD-Ausgabe



1. LCD-Ausgabe

Anschluss des LC-Displays

serielle Verbindung, genutzt werden nur drei Drähte



TxD1 Sendesignal USART1
 RxD Empfangssignal LCD

VCC Versorgungsspannung
 GND Masse (Ground)



Konfiguration von USART1 auf CEREBOT

Mit den Jumper-Einstellungen wie auf dem Bild ist im Mikrokontroller 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:

UCSR1C = 0b00000110;



1. LCD-Ausgabe

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

Name	Address	Value	Bits	
UBRR1H	0x98	0x00	<input type="checkbox"/>	} Baudrate 9600
UBRR1L	0x99	0x31	<input checked="" type="checkbox"/>	
UCSR1B	0x9A	0x08	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
RXCIE1		0x0C	<input type="checkbox"/>	
TXCIE1		0x0C	<input type="checkbox"/>	
UDRIE1		0x0C	<input type="checkbox"/>	
RXEN1		0x0C	<input type="checkbox"/>	Empfang aktivieren ¹
TXEN1		0x01	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Senden aktiviert
UCSZ12		0x0C	<input type="checkbox"/>	
RXB81		0x0C	<input type="checkbox"/>	
TXB81		0x0C	<input type="checkbox"/>	
UCSR1A	0x9B	0x20	<input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
UDR1	0x9C	0x00	<input type="checkbox"/>	
UCSR1C	0x9D	0x06	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	
UMSEL1		0x0C	<input type="checkbox"/>	asynchron Übertragung
UPM1		0x0C	<input type="checkbox"/>	kein Paritätsbit
USBS1		0x0C	<input type="checkbox"/>	1 Stoppbit
UCSZ1		0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	8 Datenbits
UCPOL1		0x0C	<input type="checkbox"/>	

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.

```

#include ...//Header einbinden
int main(){
// Vereinbarung lokaler Variablen
// * Zeichenkette mit Ausgabertext
// * 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

```

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



1. LCD-Ausgabe

Scenshoot 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         // adressiertes 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 Unterbrechungspunkt, IO-View USART-Einstellungen kontrollieren.
- Einzelschritt, Locals (lokale Variablen) Test und Zeigerwert kontrollieren.
- Programmfortsetzung bis Unterbrechungspunkt. In »Locals« 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;
    ...
}
```

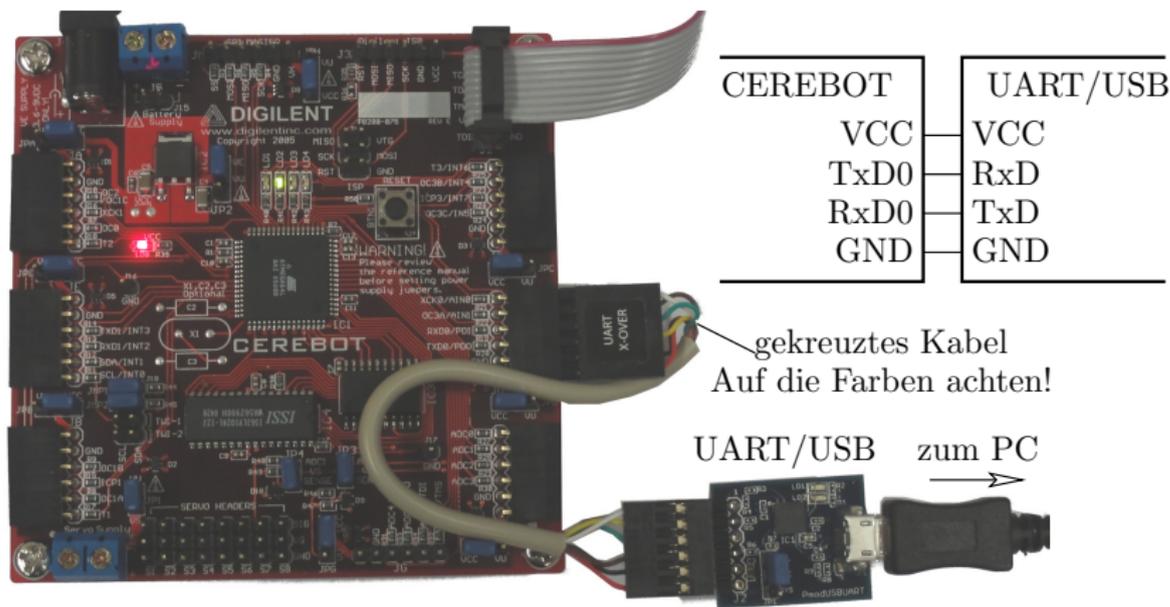


PC-Verbindung über USART/USB-Converter



2. 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	<input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>	Baudrate 9600
UCSR0B	0x2A	0x18	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Send., Empf. ein
UCSR0A	0x2B	0x20	<input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
UDR0	0x2C	0x00	<input type="checkbox"/>	
UBRR0H	0x90	0x00	<input type="checkbox"/>	Baudrate 9600
UCSR0C	0x95	0x3E	<input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	
UMSEL0		0x00	<input type="checkbox"/>	asynchron
UPM0		0x03	<input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	ungerade Parität
USBS0		0x01	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	2 Stoppbit
UCSZ0		0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	8 Datenbit
UCPOLO		0x00	<input type="checkbox"/>	



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;
    UCSRB = 0b00011000;
    UCSRC = 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.

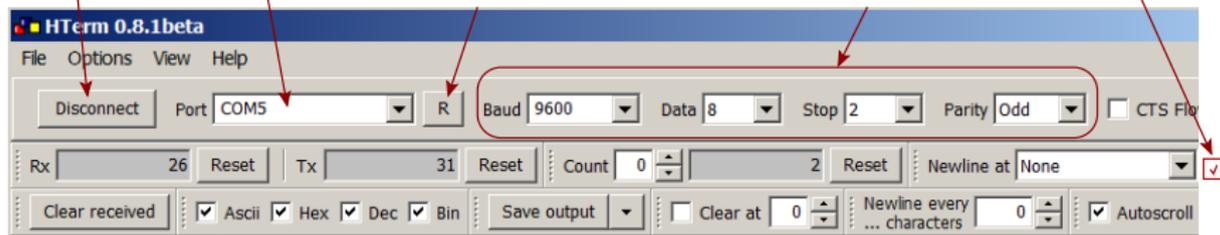
Verbindung herstellen

Schnittstelle auswählen

Schnittstellenliste aktualisieren

Einstellung des Übertragungsformats wie beim Mikrorechner

Show Errors



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



Schicken eines Textes

Received Data

1	2	3	4	5	6	7	8
I	b	m	m	p	□		
49	62	6D	6D	70	01		
073	098	109	109	112	001		
01001001	01100010	01101101	01101101	01110000	00000001		

Selection (-)

Input control

Input options

Clear transmitted | Ascii Hex Dec Bin | Send on enter Null | Send file

Type ASC | Halo ← Sendetext | Abschlussnull

Transmitted data

1	2	3	4	5	6	7	8
H	a	l	l	o	␣		
48	61	6C	6C	6F	00		
072	097	108	108	111	000		
01001000	01100001	01101100	01101100	01101111	00000000		

um 1 erhöhte Werte

rot hier würde Protokollfehler bedeuten



Sendedaten mit gemischter Zahlendarstellung

Input control

Input options

Clear transmitted | Ascii Hex Dec Bin | Send on enter None

Type BIN | **hex.dezimal binär**
A3 14573 01101101

Transmitted data: $56 \cdot 256 + 237 = 14573$

	1	2	3	4	5	6
ASCII	□	8	□	m		
Hex	A3	38	ED	6D		
Dec	163	056	237	109		
Bin	10100011	00111000	11101101	01101101		

- 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:

Input options

Clear transmitted | Ascii Hex Dec Bin | Send on enter: Null | Send file | DTR | RTS

Type: ASC | Hallo wait=1000 Welt, wait=1000 hier bin wait=1000 ich!

Transmitted data

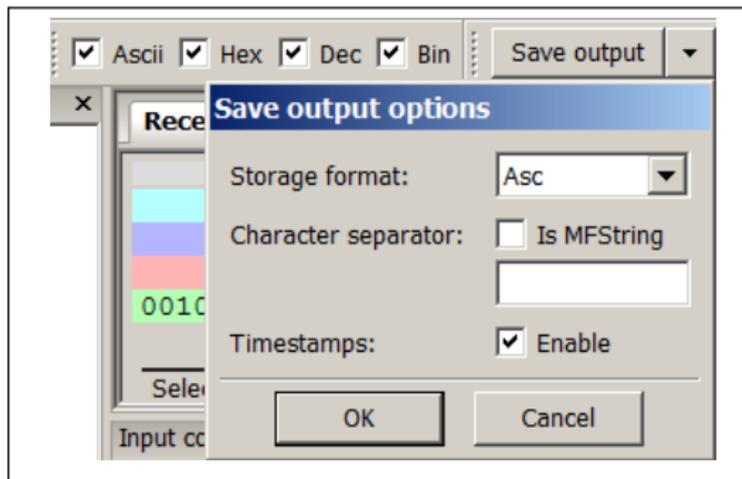
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
H	a	l	l	o	W	e	l	t	,	h	i	e	r		b	i	n	i	c	h	!	␣
48	61	6C	6C	6F	57	65	6C	74	2C	68	69	65	72	20	62	69	6E	69	63	68	21	00
072	097	108	108	111	087	101	108	116	044	104	105	101	114	032	098	105	110	105	099	104	033	000



2. PC-Verbindung über USART/USB-Converter

Die empfangenen Daten mit Zeitstempel in Datei speichern⁴.

Ausgabeformat: Ascii mit Zeitstempel



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:
!
```

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

⁴Im 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 PModUSBUSART und des PCs wie auf Folie 17.
- Anlegen eines neuen 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 gemischten Zahlen/Zeichen-Folge wie auf Folie 22.
- Test mit Warteanweisungen wie auf Folie 23 und Dateiausgabe mit Zeitstempel.
- Änderungen der Übertragungsparameter und Wiederholung einiger Tests. 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, das 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«. Probieren Sie einige der Steuerbefehle, z.B. Bildschirm löschen, Cursor bewegen, Scollen, Hintergrundbeleuchtung ein-/ausschalten etc. ⁵.
- Schreiben Sie ein Programm mit selbst definierter Zielfunktion, in dem Sie das Gelernte über die LCD-Ausgabe und die Kommunikation mit dem PC anwenden.

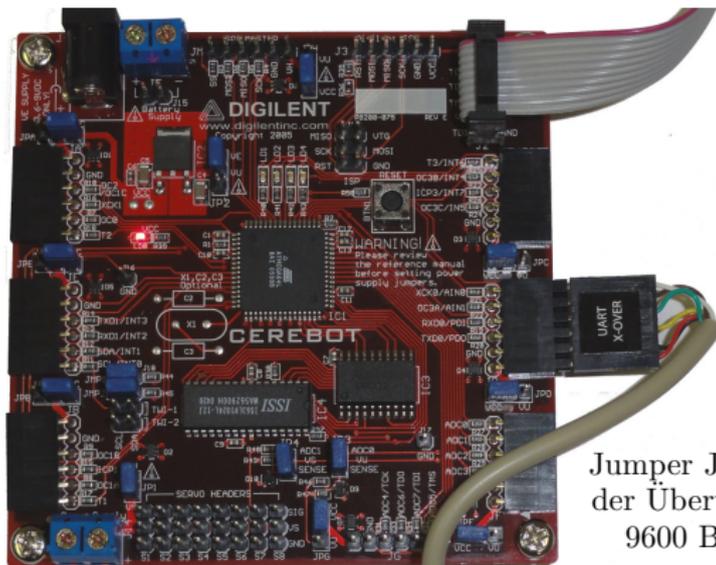
⁵Bitte ohne vorherige Rücksprache mit dem Übungsleiter keine Änderungen im EEPROM vornehmen.



Drahtlose Kommunikation über Bluetooth



Bluetooth-Modul anschließen



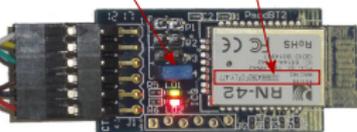
USB-Bluetooth-Dongle



Jumper JP4 zur Einstellung
der Übertragungsparameter
9600 Baud 8n1 stecken

MAC-Nummer

Bluetooth-Modul





Bluetooth-Verbindung auf PC einrichten



Select a device to add to this computer

Windows will continue to look for new devices and display them here.



RN42-1C83
Bluetooth
Other

letzte 4 Ziffern der MAC-Nummer
auf dem Bluetooth-Modul, mit dem
sich der Rechner verbinden soll



RN42-92FC
Bluetooth
Other

- Unter Windows Doppelklick auf Bluetooth-Symbol.
- Bei Bedarf »add Device«.
- Device mit der MAC-Nummer auf dem PMOD auswählen.
- Für »Enter the Device Pairing Code« Eingabe »1234«.⁶
- rechter Mouseklick > Properties > Hardware > hinterm Namen COM-Port ablesen.

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



3. Drahtlose Kommunikation über Bluetooth

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 ähnliche Tests wie bei der drahtgebundenen Kommunikation durch.