4.3 Die serielle USART-Schnittstelle

4.3.1 Serielle Datenübertragung

Die Bezeichnung **USART** ist eine Abkürzung für **U**niversal **S**ynchronous and **A**synchronous **R**eceiver and **T**ransmitter und bedeutet, dass Empfänger und Sender sowohl für die synchrone als auch für die asynchrone serielle Datenübertragung vorhanden sind. *Bild 4-36* zeigt eine *synchrone* Übertragung mit einer Datenleitung und einer Taktleitung zur Synchronisation von Sender und Empfänger.

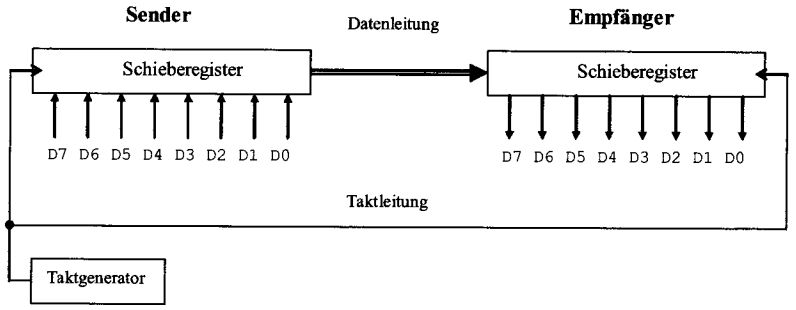

Bild 4-36: Synchrone serielle Datenübertragung mit gemeinsamer Taktleitung

Bei der in *Bild 4-37* dargestellten *asynchronen* seriellen Übertragung nach den Normen V.24 bzw. RS 232C entfällt die Taktleitung; Sender und Empfänger sind nur durch die Datenleitung und Ground (Erde) miteinander verbunden.

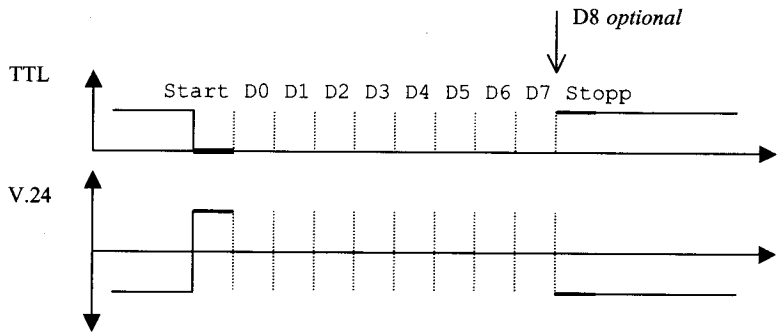

Bild 4-37: Zeitdiagramm der asynchronen seriellen Übertragung nach der Norm V.24 (RS 232C)

Die **asynchrone Schnittstelle** enthält sowohl einen Sender als auch einen Empfänger und kann im Vollduplexbetrieb gleichzeitig senden und empfangen. Die Daten werden in einen Rahmen (*frame*) aus Startbit, Stoppbit und Paritätsbit eingebettet. Da der Schiebetakt nicht übertragen wird, müssen sich Sender und Empfänger bei jedem Zeichen durch die fallende Flanke des Startbits neu synchronisieren. Ausgehend vom *Ruhezustand* TTL-High und V.24-negativ (-3 bis -15 Volt) ist das *Startbit* immer TTL-Low, und die Leitung ist V.24-positiv (+3 bis +15 Volt). Dann folgen die Datenbits. Das *Stoppbit* ist immer TTL-High und geht in den Ruhezustand über, wenn kein Zeichen direkt folgt. Zwischen dem letzten Bit D7 und dem Stoppbit kann optional ein Paritätsbit oder ein zweites Stoppbit eingeschoben werden. Die Anzahl der Übertragungsschritte pro Sekunde wird in der Einheit *baud* angegeben. Da mit jedem Schritt (Takt) ein Bit übertragen wird, ist die Baudrate gleich der Datenübertragungsrate in der Einheit *bps* (Bit pro Sekunde). Wegen der fehlenden Taktverbindung müssen Sender und Empfänger auf die gleiche Baudrate eingestellt sein, die nicht mehr als 2% von dem genormten Wert abweichen sollte. Bei 9600 baud (bps) beträgt die Bitzeit 104 µs. Die Übertragung eines Zeichens (Startbit, acht Datenbits und ein Stoppbit) dauert ca. 1 ms. Mit der USART-Schnittstelle lässt sich eine Verbindung zur seriellen Schnittstelle des PC (COM) herstellen und mit einem Terminalprogramm wie z.B. „HyperTerminal" testen. Der Pegelwandler MAX 232 *Bild 4-38* passt die Controllersignale an die COM-Schnittstelle an.

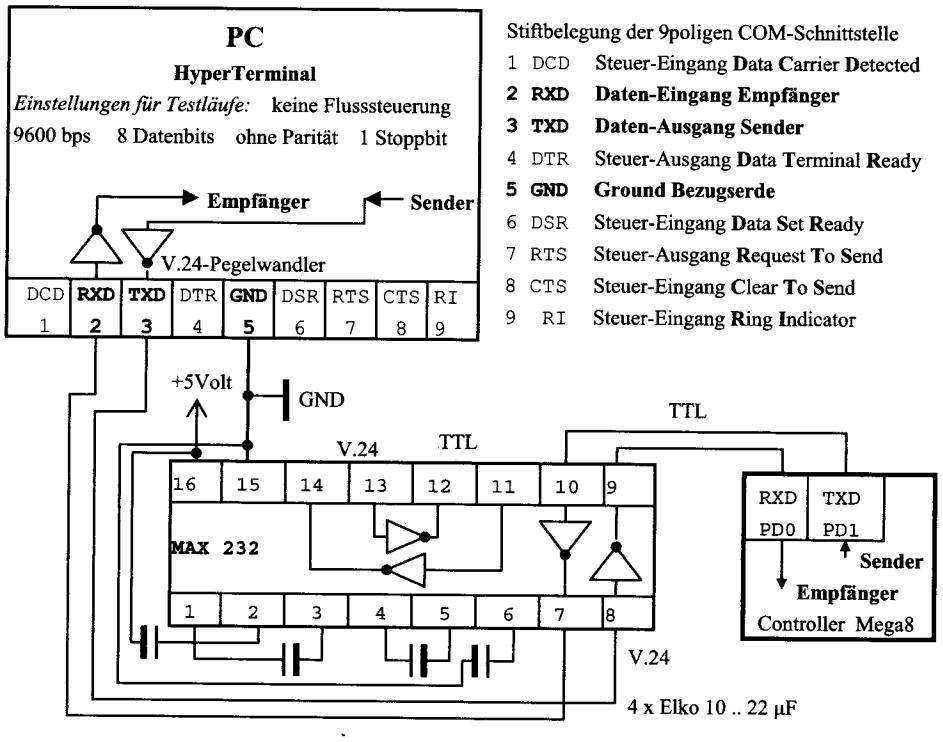

Bild 4-38: Test der USART-Schnittstelle mit TTL/V.24-Pegelwandler MAX 232 und HyperTerminal

Die COM-Schnittstelle des PC ist für den Anschluss eines Modems (Modulator/Demodulator) mit besonderen Steuersignalen eingerichtet, die einen Quittungsbetrieb (Handshake) ermöglichen. Die USART-Schnittstelle muss diese Modemsignale – wenn erforderlich – mit zusätzlichen Leitungen der Parallelports übertragen. *Bild 4-39* zeigt links eine Drei-Draht-Verbindung mit einem 1:1-Kabel und einem V.24/TTL-Pegelwandler sowie rechts eine Null-Modem-Schaltung mit gekreuzten Verbindungen ohne einen Pegelwandler, der für kurze Leitungen (ca. 1 m) zwischen zwei Controllern nicht erforderlich ist.

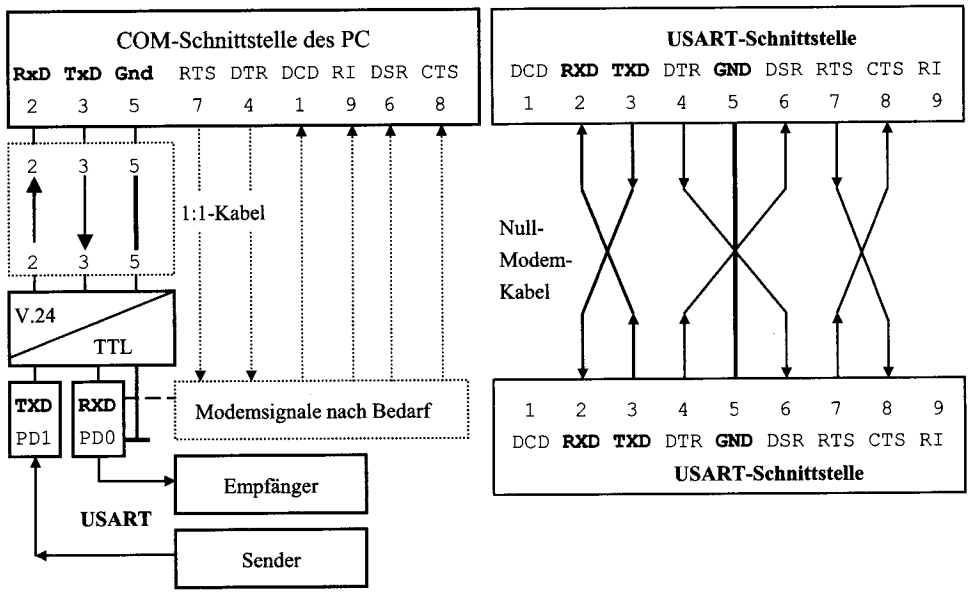

Bild 4-39: Drei-Draht-Verbindung mit 1:1-Kabel und Null-Modem-Kabel (9polige Anschlüsse)

Für den Test der folgenden Beispielprogramme ist neben dem in *Bild 4-38* dargestellten Pegelwandler auf dem PC ein Terminalprogramm erforderlich, das von der PC-Tastatur eingegebene Zeichen an die USART-Schnittstelle sendet und von der Schnittstelle ankommende Zeichen auf dem PC-Bildschirm darstellt. Das Windows-Programm HyperTerminal wurde eingestellt für 9600 bps, 8 Datenbits, 1 Stoppbit, ohne Parität und ohne Flusssteuerung. Abschnitt 4.3.6 behandelt die drei Übertragungsverfahren der Flusssteuerung:

Ohne Flusssteuerung: Sender immer freigegeben.

Flusssteuerung Xon/Xoff: Freigabe des Senders durch Steuerzeichen.

Flusssteuerung Hardware: Freigabe des Senders durch eine Signalleitung.

4.3.2 Der Asynchronbetrieb der USART-Schnittstelle

Die Sende- und Empfangsdaten werden in Doppelpuffern zwischengespeichert. Die Datenübertragung erfolgt über das Datenregister UDR, das beim Schreiben die Sendedaten aufnimmt und beim Lesen die empfangenen Daten enthält.

UDR = **U**SART **D**ata **R**egister bitadressierbar

schreiben:	Sendedaten nach Senderpuffer übertragen
lesen:	Empfangsdaten aus Empfängerpuffer abholen

Sender und Empfänger der Schnittstelle werden für die gleiche Baudrate programmiert. Für die einfache asynchrone Übertragungsgeschwindigkeit (Bit U2X = 0 in UCSRA) ergeben sich der ganzzahligeTeiler, die tatsächliche Baudrate und der Fehler aus den Formeln:

$$\text{Teiler} = \frac{\text{Systemtakt}}{16 * \text{Baud}} - 1 \qquad \textbf{z.B.}\ \frac{8\ \text{MHz}}{16 * 9600} - 1 = 51.08 \text{ gerundet } 51$$

$$\text{Baud} = \frac{\text{Systemtakt}}{16 * (\text{Teiler} + 1)} \qquad \textbf{z.B.}\ \frac{8\ \text{MHz}}{16*(51 + 1)} = 9615.4 \text{ tatsächlich}$$

$$\text{Fehler} = \frac{\text{Baudrate}_{\text{tatsächlich}}}{\text{Baudrate}_{\text{genormt}}} - 1 * 100 \qquad \textbf{z.B.}\ \frac{9615.4}{9600} - 1 * 100 = 0.16\%$$

Bei verdoppelter asynchroner Übertragungsgeschwindigkeit (Bit U2X = 1 in UCSRA) ist der Faktor 16 im Nenner durch den Faktor 8 zu ersetzen. Im Synchronbetrieb (Bit UMSEL = 1) ist der Faktor 16 im Nenner durch den Faktor 2 zu ersetzen.

Für die einfache Baudrate (Faktor 16 im Nenner) ergibt sich mit einer Nachpunktstelle:

Baud	*1 MHz*	*2 MHz*	*3.2768 MHz*	*3.6864 MHz*	*4 MHz*	*7.3728 MHz*	*8 MHz*	*16 MHz*
2400	25.0	51.1	84.3	95	103.2	191.0	207.3	415.7
4800	12.0	25.0	42.7	47	51.1	95.0	103.2	207.3
9600	***5.5***	12.0	20.3	23	25.0	47.0	51.1	103.2

Für einen Systemtakt von 1 MHz und die genormte Baudrate von 9600 liefert der gerundete Teiler 5 eine einfache Baudrate von 10417. Dies ergibt einen Fehler von 8.5%! Die folgenden Programmbeispiele arbeiten daher mit doppelter Baudrate (Faktor 8 im Nenner) und einem Teiler von 12.02 gerundet 12 bei einem Fehler von 0.16%.

Das Low-Byte der 12 bit langen ganzzahligen Baudrate ist nach UBRRL zu schreiben.

UBRRL = USART **B**aud **R**ate **R**egister **L**ow bitadressierbar

Baudrate Bit 7 bis Bit 0

Das High-Byte ist nach UBRRH zu schreiben, das nach einem Reset zunächst gelöscht ist. Da es auf der gleichen Adresse wie das Steuer- und Statusregister UCSRC liegt, muss beim Zugriff auf das Baudratenregister das Umschaltbit URSEL = 0 sein.

UBRRH = USART **B**aud **R**ate **R**egister **H**igh

Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
URSEL = 0	–	–	–	Baudrate Bit 11 bis Bit 8			

Nach einem Reset ist die USART-Schnittstelle zunächst gesperrt. Sie muss durch Programmierung der drei Steuerregister freigegeben werden, die als Statusregister gleichzeitig auch Anzeige- und Zustandsbits enthalten.

UCSRA = USART Control Status **R**egister **A** bitadressierbar

Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
RXC	**TXC**	**UDRE**	**FE**	**DOR**	**PE**	**U2X**	**MPCM**
Empfänger 0: kein Zei. 1: Zeichen *Interrupt*	*Daten aus Schiebereg.* 1: gesendet *Interrupt*	*Sendedaten-Register* 1: leer *Interrupt*	*Empfangs-Rahmen* 1: Fehler	*Empfänger-Überlauf* 1: Fehler	*Paritäts-kontrolle* 1: Fehler	*Doppelte Baudrate* 0: Faktor 16 1: Faktor 8	*Steuerbit Multi-prozessor-betrieb*

Das Anzeigebit **RXC** wird von der Steuerung auf 1 gesetzt, wenn ein Zeichen im Empfänger angekommen ist und wird beim Lesen der Daten aus UDR wieder gelöscht.

Das Anzeigebit **TXC** wird von der Steuerung auf 1 gesetzt, wenn ein Zeichen aus dem Sender herausgeschoben wurde und der Sendepuffer leer ist. Es wird bei der Interruptannahme bzw. durch Einschreiben einer 1 wieder gelöscht.

Das Anzeigebit **UDRE** wird von der Steuerung auf 1 gesetzt, wenn der Sendepuffer für die Übertragung neuer Daten bereit ist. Nach einem Reset ist der Sender bereit und UDRE ist 1.

Die Anzeigebits **FE**, **DOR** und **PE** werden von der Steuerung auf 1 gesetzt, wenn ein entsprechender Fehler aufgetreten ist. Sie werden durch Lesen des Datenregisters wieder auf 0 zurückgesetzt.

Das Steuerbit **U2X** = 1 (Double the USART Transmission Speed) verdoppelt mit einer 1 die Baudrate im asynchronen Betrieb. Dann ist im Nenner der Formel für den Taktteiler der Faktor 16 durch den Faktor 8 zu ersetzen!

Das Steuerbit **MPCM** = 1 (**M**ulti-**P**rocessor **C**ommunication **M**ode) schaltet für den Empfänger den Multiprozessorbetrieb ein, in dem bei empfangenen Zeichen zwischen Daten und Adressen unterschieden wird.

UCSRB = USART **C**ontrol **S**tatus **R**egister **B** bitadressierbar

Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
RXCIE	**TXCIE**	**UDRIE**	**RXEN**	**TXEN**	**UCSZ2**	**RXB8**	**TXB8**
Empfänger Interrupt 1: frei	*Sender Interrupt* 1: frei	*Sendedaten Interrupt* 1: frei	*Empfänger* 0: gesperrt 1: frei	*Sender* 0: gesperrt 1: frei	*Zeichenlänge siehe* **UCSRC**	*Empfänger Bit 8*	*Sender Bit 8*

Das Steuerbit **RXCIE** ist vom Programm auf 1 zu setzen, wenn bei einem empfangenen Zeichen (`RXC = 1`) und für `I = 1` im Statusregister ein Interrupt ausgelöst werden soll.

Das Steuerbit **TXCIE** ist vom Programm auf 1 zu setzen, wenn bei einem herausgeschobenen Zeichen (`TXC = 1`) und für `I = 1` im Statusregister ein Interrupt ausgelöst werden soll.

Das Steuerbit **UDRIE** ist vom Programm auf 1 zu setzen, wenn bei einem leeren Sendedatenregister (`UDRE = 1`) und für `I = 1` im Statusregister ein Interrupt ausgelöst werden soll.

Die Einsprünge der drei Interrupt-Serviceprogramme liegen im unteren Adressbereich und werden von den meisten Definitionsdateien als Symbole vereinbart.

- Empfängerinterrupt: Assembler `URXCaddr` und für C: `SIG_UART_RECV`
- Sendeschieberegisterinterrupt: Assembler `UTXCaddr` und für C: `SIG_UART_TRANS`
- Sendedatenregisterinterrupt: Assembler `UDREaddr` und für C: `SIG_UART_DATA`

```
; Assemblerbeispiel für einen Empfängerinterrupt
     .ORG   URXCaddr     ; Einsprungadresse
     rjmp   abholen      ; Sprung zum Serviceprogramm
```

```
// C-Servicefunktion für einen Empfängerinterrupt
SIGNAL(SIG_UART_RECV)
{  /* Zeichen abholen */  }
```

Das Steuerbit **RXEN** ist vom Programm auf 1 zu setzen, um den Empfänger einzuschalten. Die Portfunktionen sind dabei abgeschaltet.

Das Steuerbit **TXEN** ist vom Programm auf 1 zu setzen, um den Sender einzuschalten. Die Portfunktionen sind dabei abgeschaltet.

Das Steuerbit **UCSZ2** (**U**SART **C**haracter **SiZe**) bestimmt zusammen mit UCSZ1 und UCSZ0 des Steuerregisters UCSRC die Anzahl der Datenbits von Sender und Empfänger.

Die Bitposition **RXB8** enthält bei einer Übertragung von neun Datenbits die neunte Bitposition der empfangenen Daten und muss vor dem Lesen des Datenregisters UDR gelesen werden.

In die Bitposition **TXB8** ist bei einer Übertragung von neun Datenbits die neunte Bitposition zu schreiben bevor die restlichen acht Bitpositionen nach UDR geschrieben werden.

Das Steuer- und Statusregister UCSRC liegt auf der gleichen Adresse wie das Baudratenregister UBRRH und muss mit dem Steuerbit URSEL = 1 eingeschaltet werden.

UCSRC = **U**SART **C**ontrol **S**tatus **R**egister **C**

Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
URSEL	**UMSEL**	**UPM1**	**UPM0**	**USBS**	**UCSZ1**	**UCSZ0**	**UCPOL**
URSEL = 1	*Betriebsart* 0: *Async.* 1: *Sync.*	*Parität:* 0 0: keine 0 1: reserviert 1 0: gerade Parität 1 1: ungerade Parit.		*Stoppbits* 0: 1 Bit 1: 2 Bits	*Datenlänge* UCSZ2 UCSZ1 UCSZ0 0 0 0: 5bit Übertragung 0 0 1: 6bit Übertragung 0 1 0: 7bit Übertragung 0 1 1: 8bit Übertragung 1 1 1: 9bit Übertragung		*Synchronbetrieb Phasenlage des Schiebetaktes*

Das Steuerbit **UMSEL** (**U**SART **M**ode **SEL**ect) schaltet mit einer 0 den asynchronen und mit einer 1 den synchronen Betrieb ein.

Die Steuerbits **UPM1** und **UPM2** (**U**SART **P**arity **M**ode) wählen die Parität der übertragenen Daten aus, die mit dem Anzeigebit PE auf Paritätsfehler überprüft wird.

Das Steuerbit **USBS** (**U**SART **S**top **B**it **S**elect) legt die Anzahl der Stoppbits für die zu sendenden Daten fest. Die Angabe ist für den Empfänger, der nur ein Stoppbit benötigt, wirkungslos.

Die Steuerbits **UCSZ1** und **UCSZ0** (**U**SART **C**haracter **SiZe**) legen zusammen mit UCSZ2 die Anzahl der übertragenen Datenbits fest.

Das Steuerbit **UCPOL** (**U**SART **C**lock **POL**arity) bestimmt nur im Synchronbetrieb die Phasenlage der gesendeten bzw. empfangenen Datenbits in Bezug auf den Übertragungstakt. Im Asynchronbetrieb sollte UCPOL = 0 sein. Im synchronen Sendebetrieb werden für das Bit UCPOL = 0 die an TxD gesendeten Datenbits mit der steigenden Flanke des Sendetaktes XCK ausgegeben, für UCPOL = 1 mit der fallenden Flanke. Im synchronen Eingabebetrieb werden für UCPOL = 0 die an RxD ankommenden Datenbits mit der fallenden Flanke des Taktes XCK abgetastet, für UCPOL = 1 mit der steigenden Flanke.

Die *C-Funktionen* sind mit ihren `#include`-Direktiven in der Headerdatei `konsole.h` zusammengefasst.

```
// konsole.h  Headerdatei Konsolfunktion USART
#include   "initusart.c"  // USART init. einfache Baudrate
#include   "initusart2.c" // USART init. doppelte Baudrate
#include   "putch.c"      // Zeichen nach Sender
#include   "getch.c"      // warten und Zeichen von Empfänger
#include   "getche.c"     // warten Zeichen von Empfänger Echo
#include   "kbhit.c"      // ohne warten Rückgabe Zeichen oder Null
#include   "putstring.c"  // String ausgeben
#include   "getstring.c"  // String lesen
#include   "cmpstring.c"  // Strings vergleichen 1:gleich 0:ungleich
```

Die Funktion `initusart` initialisiert die Schnittstelle für den Asynchronbetrieb mit acht Datenbits und einem Stoppbit ohne Parität. Es ist zweckmäßig, durch eine Handrechnung zu überprüfen, ob der gerundete Wert der Baudrate nicht mehr als 2% vom genormten Nennwert abweicht. Mit verdoppelter Baudrate (`U2X = 1`) lassen sich mit der Funktion `initusart2` gegebenenfalls günstigere Werte erzielen.

```
// initusart.c  USART initialisieren einfache Baudrate
void initusart(void)                    // USART initialisieren
{
 unsigned char x;                       // Hilfsvariable
 UBRRL = (TAKT / (16ul * BAUD)) - 1;    // Baudrate mit TAKT und BAUD
 UBRRH = 0;                             //
 UCSRA |= (0 << U2X);                   // U2X einfache Baudrate * 16
 UCSRB |= (1 << TXEN) | (1 << RXEN);    // Sender und Empfänger ein
 UCSRC |= (1 << URSEL) | (1 << UCSZ1) | (1 << UCSZ0); // async 8bit
 x = UDR;                               // Empfänger leeren
}

// initusart2.c  USART initialisieren doppelte Baudrate
void initusart2(void)                   // USART initialisieren
{
 unsigned char x;                       // Hilfsvariable
 UBRRL = (TAKT / (8ul * BAUD)) - 1;     // Baudrate mit TAKT und BAUD
 UBRRH = 0;                             //
 UCSRA |= (1 << U2X);                   // U2X doppelte Baudrate * 8
 UCSRB |= (1 << TXEN) | (1 << RXEN);    // Sender und Empfänger ein
 UCSRC |= (1 << URSEL) | (1 << UCSZ1) | (1 << UCSZ0); // async 8bit
 x = UDR;                               // Empfänger leeren
}
```

Die Funktion putch wartet bis der Sender frei ist und übergibt dann das auszugebende Zeichen dem Sendedatenregister. Bei 9600 Baud und einer Bitzeit von ca. 100 µs beträgt die maximale Wartezeit bei einem Startbit, acht Datenbits und einem Stoppbit etwa 1 ms.

```
// putch.c Zeichenausgabe für USART und UART
void putch (unsigned char x)        // warten und Zeichen senden
{
 while( ! (UCSRA & (1 << UDRE)));  // warte solange Sender besetzt
 UDR = x;                          // Zeichen nach Sender
}
```

Die Funktionen getch und getche warten auf ein ankommendes Zeichen. Die Wartezeit beträgt bei 9600 Baud und 10 Bits mindestens eine Millisekunde. Bei sehr langsamer Eingabe durch die Sendestation kann es zweckmäßig sein, die Wartezeit durch einen Empfängerinterrupt zu verkürzen. Die Funktion kbhit testet den Empfänger und kehrt mit dem Rückgabewert Null zurück, wenn kein Zeichen angekommen ist. Die Beispiele verzichten auf die Auswertung der Fehlermarken für Überlauf, Rahmen und Parität.

```
// getch.c Zeichen von USART holen
unsigned char getch(void)           // warten und Zeichen abholen
{
 while ( ! (UCSRA & (1 << RXC)));  // warte bis Zeichen da
 return UDR;                       // Zeichen abholen
}

// getche.c Eingabe mit Echo von USART
unsigned char getche(void)          // warten und lesen mit Echo
{
 int x;                            // Hilfsvariable
 while ( ! (UCSRA & (1 << RXC)));  // warte bis Zeichen da
 x = UDR;                          // abholen und speichern
 while( ! (UCSRA & (1 << UDRE)));  // warte solange Sender besetzt
 UDR = x;                          // Echo senden
 return x;                         // Zeichen zurückgeben
}


// kbhit.c kein Zeichen: Rückgabe 0 sonst Rückgabe Zeichen
unsigned char kbhit(void)           // Empfänger testen
{
 if (UCSRA & (1 << RXC)) return UDR; else return 0;
}
```

Das C-Programm *Bild 4-42* testet die USART-Zeichenfunktionen mit einem PC als Terminal. Nach dem Senden des Promptzeichens **>** werden alle vom PC gesendeten Zeichen im Echo wieder zurückgeschickt. Von den mit `#include "konsole.h"` eingefügten Funktionen werden `getch` und `kbhit` nicht verwendet. Die in der Hauptfunktion `main` definierten Symbole `TAKT` und `BAUD` werden für die Initialisierung in der Funktion `initusart2` benötigt.

```
// k4p14.c  ATmega8  USART Test der Konsolfunktionen
// Port B: -
// Port D: PD0 -> RXD  PD1 -> TXD
// Konfiguration: interner Oszillator 1 MHz, externes RESET-Signal
#include <avr/io.h>          // Deklarationen
#define TAKT 1000000UL       // Controllertakt 1 MHz
#define BAUD 9600UL          // Baudrate
#define SLAENG 80            // für Funktion getstring in konsole.h
#include "konsole.h"         // Funktionen initusart2,putch,getch,getche,kbhit
main(void)                   // Hauptfunktion
{
 initusart2();               // USART initialisieren doppelte Baudrate
 putch('>');                 // Prompt ausgeben
 while(1)                    // Arbeitsschleife
 {
  getche();                  // Zeichen lesen mit Echo
 } // Ende while
} // Ende main
```

Bild 4-42: C-Programm zum Testen der USART-Zeichenfunktionen