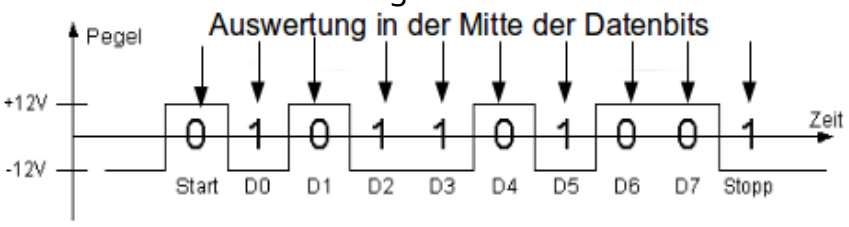
Serielle Schnittstelle

RS-232 ist ein Standard für eine bei Computern teilweise vorhandene serielle Schnittstelle, der in den frühen 1960er Jahren von dem US-amerikanischen Standardisierungskomitee Electronic Industries Association (EIA) erarbeitet wurde. (wikipedia WP)

- Die Übertragung erfolgt in Wörtern
  - meistens erfolgt die Kodierung gemäß ASCII. Üblich ist daher, sieben bzw. acht Datenbits zu übertragen.
  -

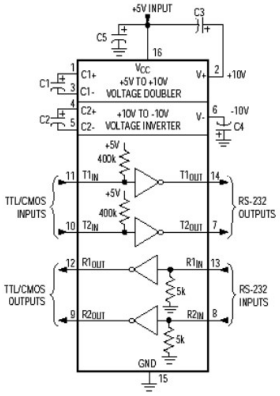
Dez	Hex	Okt	ASCII	Dez	Hex	Okt	ASCII	Dez	Hex	Okt	ASCII	Dez	Hex	Okt	ASCII
0	0x00	000	NUL	32	0x20	040	SP	64	0x40	100	@	96	0x60	140	`
1	0x01	001	SOH	33	0x21	041	!	65	0x41	101	A	97	0x61	141	a
2	0x02	002	STX	34	0x22	042	"	66	0x42	102	B	98	0x62	142	b
3	0x03	003	ETX	35	0x23	043	#	67	0x43	103	C	99	0x63	143	c
4	0x04	004	EOT	36	0x24	044	$	68	0x44	104	D	100	0x64	144	d
5	0x05	005	ENQ	37	0x25	045	%	69	0x45	105	E	101	0x65	145	e
6	0x06	006	ACK	38	0x26	046	&	70	0x46	106	F	102	0x66	146	f
7	0x07	007	BEL	39	0x27	047	'	71	0x47	107	G	103	0x67	147	g
8	0x08	010	BS	40	0x28	050	(	72	0x48	110	H	104	0x68	150	h

- Eine RS-232-Verbindung arbeitet *(bit-)seriell* mit je einer Datenleitung für beide Übertragungsrichtungen. Das heißt, die Bits werden nacheinander auf einer Leitung übertragen, im Gegensatz zur parallelen Datenübertragung. Die dafür nötige Seriell-Parallel-Wandlung geschieht meistens in sog. UARTs (entweder als integriertes Modul in einem Mikrocontroller oder als Einzelbaustein).
- Die Datenübertragung erfolgt *asynchron*, es existiert also kein gemeinsamer Takt.Die Synchronisation in der Übertragung erfolgt durch den Empfänger als sogenannte Wortsynchronisation, also am Anfang durch die Signalflanke des Startbits.
- Vereinbarung der Baudrate (=Anzahl der Symbole / Sekunde) zwischen Sender und Empfänger (9600, 19200, ... 115200)
- Die Synchronisation des Empfängers geschieht mit dem Start der Übertragung auf der Datenleitung, da das Stopp-Bit bzw. der Ruhezustand auf der Leitung den inversen Pegel zum Start-Bit aufweist. Der Empfänger synchronisiert sich so in die Mitte der einzelnen Datenbits und tastet die folgenden Bits des Datenwortes mit seiner eigenen Bitrate ab.



- Damit das funktioniert, dürfen die Bitraten von Sender und Empfänger nur einige Prozent voneinander abweichen (ca. 3%).
- Jedes übertragene Wort muss somit von einem Startbit (logischer Wert 0) eingeleitet und mit mindestens einem Stopp-Bit (logischer Wert 1) abgeschlossen werden.
- RS-232 ist eine *Spannungsschnittstelle* (im Gegensatz z.B. zu einer Stromschnittstelle). Die binären Zustände werden durch verschiedene elektrische Spannungspegel realisiert.
- Für die Datenleitungen (TxD und RxD) wird eine negative Logik verwendet, wobei eine Spannung zwischen −3 V und −15 V eine logische Eins bedeutet. Signalpegel zwischen −3V und +3V gelten als undefiniert.
- Bei den Steuerleitungen (DCD, DTR, DSR, RTS, CTS und RI) wird positive Logik verwendet.
- *Steckverbindung:* 9-polige D-Sub-Stecker und Buchsen
- **Handshake:** Zur Vermeidung von Datenverlusten muss der Empfänger die Datenübertragung anhalten können, wenn keine weiteren Daten mehr verarbeitet werden können.
    - softwareseitig über bestimmte Steuercodes
    - oder über spezielle Leitungen (Hardware-Handshake).
- Grundsätzlich ist eine Vollduplex-Verbindung möglich, da für Sendung und Empfang getrennte Datenleitungen zur Verfügung stehen.
- Kabellänge:Laut ursprünglichem Standard ist eine Kabelkapazität von max. 2500 pF zulässig, was bei Standardkabeln einer Kabellänge von max. 15 m

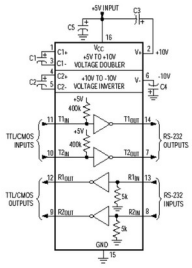
max. Baud	max. Länge
2.400	900 m
4.800	300 m
9.600	152 m
19.200	15 m
57.600	5 m
115.200	<2 m

- 

RS232 Treiber Bausteine zur Erzeugung der Hochspannung (+-10V)
- Parity; odd/even prüft, ob die Gesamtzahl der 1 inklusive Parity-Flag gerade

oder ungerade ist z.B: Sender und Empfänger vereinbaren 8Bit even Parity: das 8.Bit ist das Parity-Bit: gesendet: 1101 0010 , aber empfangen wurde 1100 0010 → liefert einen Parity-Error

- Konfiguration der Seriellen Schnittstelle 9600 8N1 (9600 Baud, 8 Bit Daten, keine Paritätsprüfung, 1 Stopbit



AVR UART

Data Register UDR

Control Register UCR

Baud Rate Register UBR

Status Register USR

Control&Status Register UCSR

Status Flags

- Receive Complete
- Transmit complete
- Data register empty
- Error

Control Register

- Rx Complete Interrupt Enable
- Tx Complete Interrupt Enable
- Receive enable
- Transmit enable

Data Register

- Physikalisch getrennte Register für Senden und Empfangen, aber beide Register heißen gleich (z.B: UDR0)

Baudrate

http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial/Der_UART#UART_initialisieren

```
#define F_CPU 4000000UL  // Definition als unsigned long beachten
                         // Ohne ergeben sich unten Fehler in der Berechnung

#define BAUD 9600UL      // Baudrate

// Berechnungen
#define UBRR_VAL ((F_CPU+BAUD*8)/(BAUD*16)-1)   // Wert für das Baudratenreg.
#define BAUD_REAL (F_CPU/(16*(UBRR_VAL+1)))     // Reale Baudrate
#define BAUD_ERROR ((BAUD_REAL*1000)/BAUD) // Fehler in Promille, 1000 = kein
Fehler.

#if ((BAUD_ERROR<990) || (BAUD_ERROR>1010))
  #error Systematischer Fehler der Baudrate grösser 1% und damit zu hoch!
#endif
```

Initialisierung

```
void uart_init(void)
{
    UBRRH = UBRR_VAL >> 8;
    UBRRL = UBRR_VAL & 0xFF;

    UCSRC = (1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0);  // Asynchron 8N1
    UCSRB |= (1<<RXEN);                        // UART RX einschalten
    UCSRB |= (1<<TXEN);                        // UART TX einschalten
}
```

Daten senden

```
/* ATmega16 */
int uart_putc(unsigned char c)
{
    while (!(UCSRA & (1<<UDRE)))  /* warten bis Senden moeglich */
    {
    }

    UDR = c;                      /* sende Zeichen */
    return 0;
}


/* puts ist unabhaengig vom Controllertyp */
void uart_puts (char *s)
{
    while (*s)
    {   /* so lange *s != '\0' also ungleich dem "String-
Endezeichen(Terminator)" */
        uart_putc(*s);
        s++;
    }
}
```

Daten empfangen

```
uint8_t uart_getc(void)
{
    while (!(UCSRA & (1<<RXC)))   // warten bis Zeichen verfuegbar
        ;
    return UDR;                   // Zeichen aus UDR an Aufrufer zurueckgeben
}
```