Serielle Schnittstelle:

RS-232:
RS-232 ist ein Standard für die serielle Schnittstelle, die bei PC eingesetzt wird.

Aktuelle Anwendung:[1]
Es werden weltweit kaum noch Geräte produziert, die eine RS-232-Schnittstelle aufweisen. Ausgenommen sind Service- und Konfigurationsanschlüsse bei diversen Geräten (z. B. Router, Switches, Speichersysteme). Moderne Bussysteme und Datenübertragungsprotokolle bieten zuverlässigere und schnellere Verbindungsmöglichkeiten. Nur noch wenige PCs werden mit einem COM-Port ausgeliefert, Notebookhersteller bieten diese Ausstattungsoption nahezu gar nicht mehr an. Zur Mikrocontroller-Programmierung oder um ältere, noch in Gebrauch befindliche Geräte, wie z. B. Telefonanlagen und Steuergeräte, mit modernen Computern zu betreiben und programmieren zu können, sind auf dem Markt Schnittstellen-Wandler von USB nach seriell erhältlich.

Der RS-232-Standard lässt auch Prozesse zu, die mit heutigen Schnittstellen prinzipbedingt nicht möglich / ansteuerbar sind, wie das simple Aufleuchten einer LED verbunden mit einer Spannungsquelle am Port und der Masse.

Einführung Grundlagen
Textquelle: http://www.sprut.de/electronic/interfaces/rs232/rs232.htm#null

Entwicklung:
Die aus dem Jahre 1962 stammende RS232-Schnittstelle ist zwar mittlerweile ein Veteran unter den seriellen Schnittstellen in der Computertechnik, aber unter Bastlern und bei industriellen Anwenden ist sie nach wie vor sehr beliebt. Die Ursache dafür sind ihre geringen Ansprüche an Hardware und Software.

Technik: RS-232 benutzt für die Datenübertragung ein **einfaches asynchrones serielles** Verfahren.

- **Seriell bedeutet**, dass die einzelnen Bits des zu übertragenden Bytes nacheinander über eine einzige Datenleitung geschoben werden.
- **Asynchron heißt**, dass es keine Taktleitung gibt, die dem Datenempfänger genau sagt, wann das nächste Bit auf der Datenleitung liegt.

So ein Verfahren kann nur funktionierten, wenn

- Sender und Empfänger mit genau dem **gleichen internen Takt** arbeiten (Baudrate), und wenn
- der Empfänger gesagt bekommt, **wann das erste Bit genau anfängt (Synchronisation)**.

Spannungspegel (für die **Programmierung nicht relevant**):

Alle RS232-Leitungen (mit Ausnahme der Masseleitung) arbeiten mit den Spannungspegeln **+12V (für eine logische '0') und -12V (für eine logische '1')**. (Erlaubt sind jeweils 5V..15V.)

Der Datenempfänger **erwartet eine Spannung von über +3V für eine 0 und von unter -3V für eine 1**.

Für die Steuerleitungen (Handshakeleitungen) bedeutet ON einen hohen Pegel (+5V .. +15V) und OFF einen negativen Pegel (-5V ... -15V).

[1] Textquelle: http://de.wikipedia.org/wiki/RS-232 [2015-01-20]

Da sich diese Spannungspegel nicht besonders gut mit den heutzutage meist verwendeten TTL-Pegeln vertragen, werden in der Regel spezielle Treiberschaltkreise eingesetzt, um diese Pegel auf TTL-Pegel anzupassen ('1' = OFF = 5V; '0' = ON = 0V). Solche Treiberschaltkreise sind z.B. der MAX232 und seine Familienangehörigen.

Um Daten minimalistisch von einer Datenquelle zu einem Datenempfänger zu übertragen, werden eigentlich nur 2 Drähte benötigt - eine Masseleitung und eine Datenleitung. Für eine bidirektionale Verbindung reichen also **3 Leitungen (Tx, Rx, GND) (Software-Handshakeleitungen).**

Um zu vermeiden, dass der Sender Daten sendet, obwohl der Empfänger noch nicht bereit ist, werden (im Nullmodemkabel) **2 bis 4 zusätzliche Hardware-Handshakeleitungen** benötigt, die weiter unten beschrieben sind.

Sampling:

Es gibt Behauptungen, dass angeblich die Takte von Sender und Empfänger nicht exakt gleich sein müssen, und dass der Empfänger das Signal mit einem Mehrfachen des Bittaktes abtastet, um sich laufend zu synchronisieren. Dem kann man so nicht ganz zustimmen. Ein solches Oversampling zur Synchronisation ist nur bei seriellen Verbindungen üblich, bei denen sich der Takt auch sicher wieder aus dem Datenstrom rekonstruieren lässt (z.B. Manchestercodierung oder CAN-Bus).

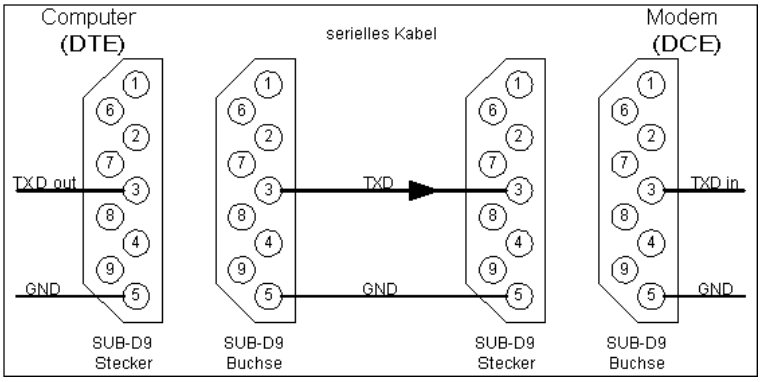
Da das bei RS-232 aber nicht gewährleistet ist, wird so etwas hier meistens auch gar nicht versucht. (Ausnahmen bestätigen natürlich auch hier die Regel.)

Allerdings wird von vielen Empfängern die Signalleitung innerhalb eines Bits schnell mehrfach hintereinander abgetastet, und dann der Pegel des Bits aufgrund einer "Mehrheitsentscheidung" bestimmt.

Eine Synchronisierung zwischen Sender und Empfänger erfolgt dabei aber nicht.

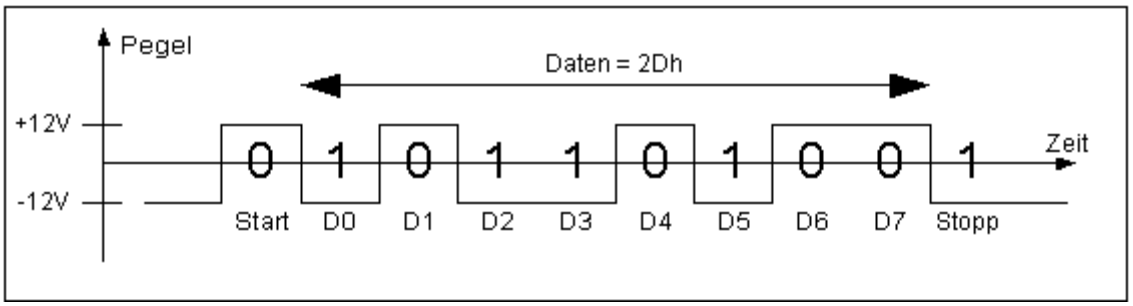
Datenleitungen

Die Daten-Ausgangs-Leitung des Senders (Computers/Terminals) wird als **TXD out** (**transmit data**; manchmal als TX) bezeichnet. Sie wird mit der Daten-Eingangs-Leitung des Empfängers (z.B. Modem) **TXD in** direkt verbunden.

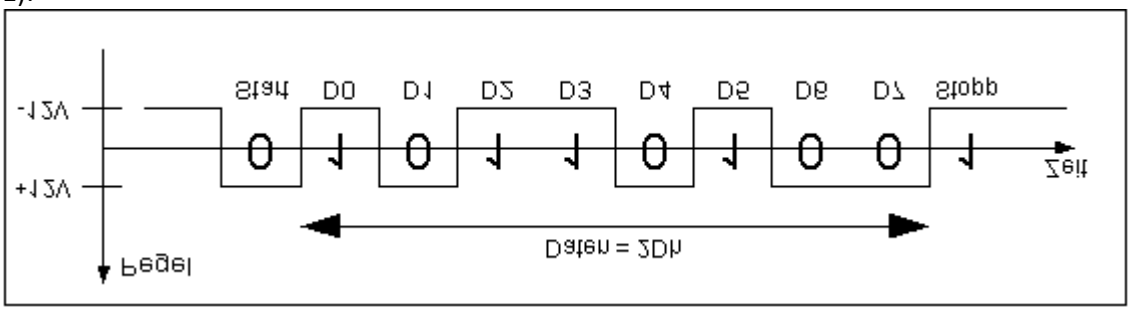


Die Datenleitung liegt im Ruhezustand auf -12V, also auf dem logischen Pegel '1'. Die Datenübertragung erfolgt byteweise (konfigurierbar auf 5-8 Bit / Byte).

Bevor ein Byte übertragen werden kann, muss der Empfänger mit dem Sender synchronisiert werden. Das erfolgt durch das Senden eines dem Datenbyte vorgesetzten Startbits mit dem Wert '0'.



Umgedreht (= 2 x Linkksdrehung, 1x Vertikalkippung) dargestellt (zur gewohnten Ansicht der Zustände 0 und 1):



Ablauf:
Am Beginn des Startbits ändert sich der Pegel der Datenleitung von -12V auf +12V. Das ist für den Empfänger das Zeichen dafür, dass der Datentransfer beginnt.

Der Sender legt nun in einem festen Zeitabstand die einzelnen Datenbits des Datenbytes auf die Datenleitung. Am Ende fügt er mindestens ein Stoppbit mit dem Wert 1 an. Damit ist die Datenleitung wieder im Ruhezustand.

Beispiel und Details:
Im oben dargestellten Bild wird die Bitfolge '10110100' übertragen, **da die Übertragung mit dem LSB (also dem letzten Bit) voran erfolgt**, ist die zeitlich hintereinander wirklich übertragene binäre Zahl: B'00101101' oder im hexadezimalen Code 0x2D bzw. $2D_h$.

Der Empfänger liest immer in der Mitte der Bits den Spannungspegel aus der Datenleitung, und empfängt so das Byte Bit für Bit.
Das funktioniert natürlich nur, wenn Sender und Empfänger mit der gleichen Geschwindigkeit die Bits schreiben und lesen. Deshalb ist es unbedingt erforderlich, dass **Sender und Empfänger vorab auf eine gleiche Datenrate eingestellt werden**. Eine Abweichung der Datenraten um mehr als 5% führt zu Lesefehlern, da der Empfänger dann bei den letzten Bits vor bzw. nach dem Bit liest.

Die Datenrate wird in **Baud (also in Bit pro Sekunde)** angegeben. Dabei werden alle Bits (auch Start- und Stopp-Bit) gezählt, und Lücken zwischen den Bytetransfers ignoriert. Deshalb ist die Baudrate der reziproke Wert der Länge eines Bits. Als Datenraten sind folgende Werte üblich:

Datenrate [baud]	300	2400	4800	9600	19200	38400	57600	115200
Bitlänge	3,33 ms	417 µs	208 µs	104 µs	52,08 µs	26,04 µs	17,36 µs	8,68 µs
maximale Kabellänge		3000 ft / 900 m	1000 ft / 300 m	500 ft / 150 m	50 ft / 15 m		15 ft / 5 m	<2 m
maximale Kabelkapazität					2500 pF			

Der Standard war ursprünglich für (nur) bis zu 19200 Baud spezifiziert worden, und **als Standardbaudrate hatte sich 9600 etabliert.**

Mit moderner Hardware ist aber inzwischen **bis zu 1.500.000 Baud technisch realisierbar**.

Die maximal erreichbare Datenrate nimmt aber auch mit der Leitungslänge ab. Ursprünglich waren maximal 17 Meter (50 Fuß) vorgesehen, und bei dieser Kabellänge sind die ursprünglich geplanten 19200 Baud erlaubt.

Deutlich längere Kabel sind machbar, wenn man die Baudrate reduziert. Dabei ermöglicht eine halbierte Datenrate eine 10 mal längere Übertragungsstrecke. Bei 2400 Baud ergibt das eine Kabellänge von 1 km. Versuch das mal mit USB :-)

Nach dem Stoppbit kann sofort wieder eine neue Übertragung mit einem Startbit beginnen. **Man kann dem Empfänger aber auch etwas mehr Zeit geben, indem man 1,5 oder 2 Stoppbits sendet.**

Falls der Empfänger seinerseits Daten zurücksenden soll, wird **dafür eine weitere Leitung benötigt**. Die Daten-Ausgangs-Leitung des Empfängers (Modems) wird als RXD out (Receive data; manchmal als RX) bezeichnet. Sie wird mit der Daten-Eingangs-Leitung des Senders (Computers) RXD direkt verbunden.

An der Namensgebung erkennt man, dass **alle Pins aus der Sicht des Computers/Terminals bezeichnet** wurden.
Auch ist die Bezeichnung Sender und Empfänger bei bidirektionalen Verbindungen nicht ideal. Besser ist es, z.B. von Computer und Modem zu sprechen. In der Praxis nennt man den schnellen Computer DTE und das langsame Modem DCE.