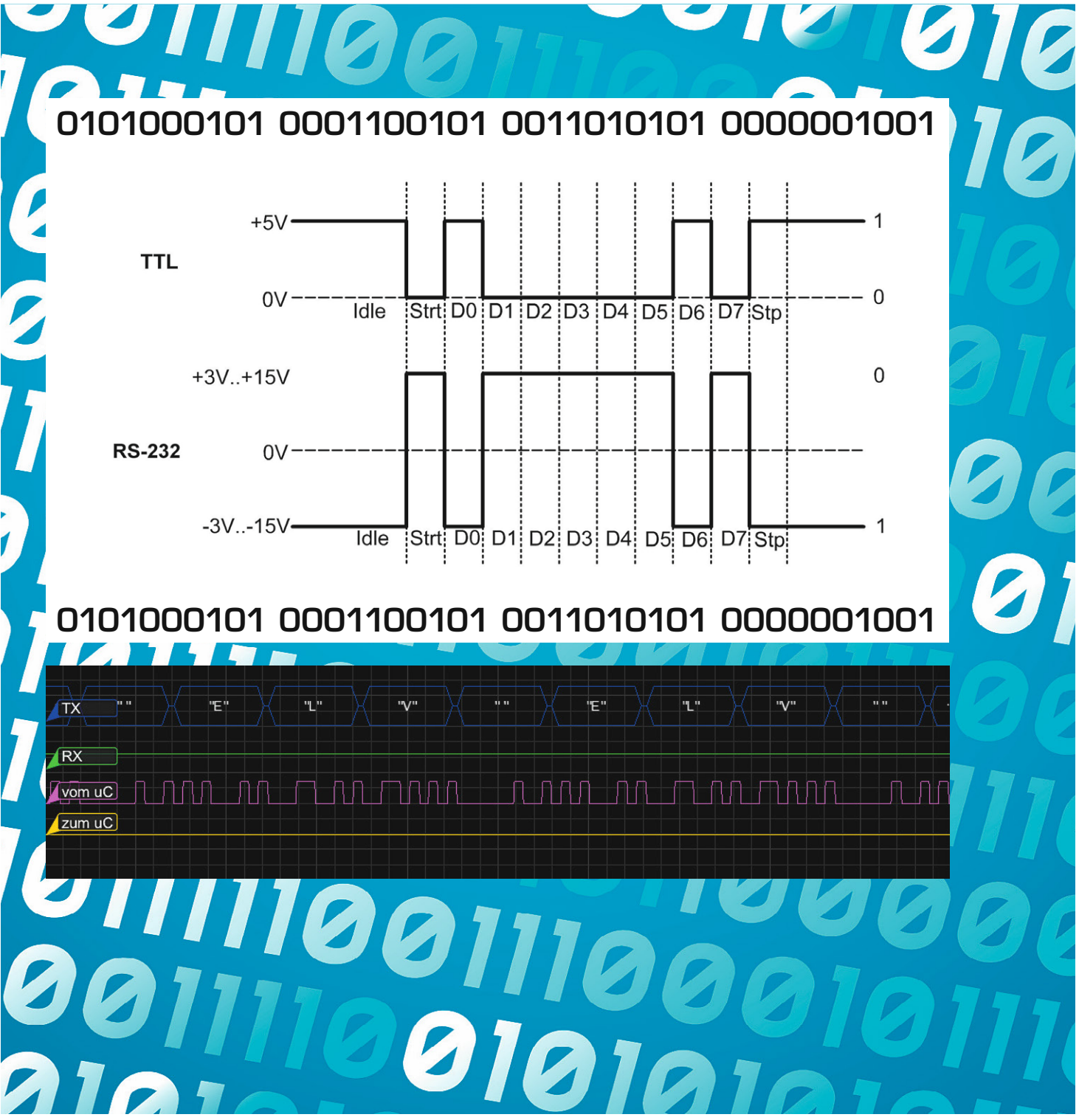
Digitale Hardwareschnittstellen

Teil 1: Die serielle Schnittstelle



In der Elektronik müssen immer wieder Systeme miteinander verbunden werden, um ein Gesamtsystem zu erstellen. Temperatursensoren werden beispielsweise per I²C oder 1-Wire angeschlossen, Echtzeituhren (Real Time Clocks) werden per I²C, SPI oder UART an eine Platine angeschlossen, GPS-Module werden über eine serielle Verbindung angeschlossen oder es wird eine Platine per USB oder seriell mit einem PC verbunden. Wer sich mit Elektronik beschäftigt, benötigt einen Überblick über die wichtigsten digitalen Hardwareschnittstellen, damit er beurteilen kann, ob verschiedene Komponenten miteinander kommunizieren können. Ein Blick in die technischen Daten eines ELV-Produkts oder in das jeweilige Datenblatt gibt Aufschluss über die jeweils unterstützte(n) Schnittstelle(n) eines Moduls oder Geräts.

In diesem Artikel werden die Grundlagen der seriellen Schnittstelle als einer der ältesten Schnittstellen, die immer noch weitverbreitet ist, dargestellt. In weiteren Artikeln werden weitere verbreitete Schnittstellen wie SPI, I²C, 1-Wire und USB beschrieben.

Schnittstellen

Die meisten Geräte bestehen aus mehreren Teilsystemen, welche man jeweils als Black Box betrachten kann und die untereinander über genau definierte Schnittstellen verbunden sind. Bild 1 zeigt schematisch ein Gesamtsystem, welches aus zwei Systemen (A und B) besteht. Ein System bietet zur Kommunikation mit anderen Systemen meistens eine oder mehrere digitale Hardwareschnittstellen an, die sehr detailliert beschrieben bzw. genormt sind. Dadurch dass die Schnittstellen genau definiert sind, können die Teilsysteme sich über eine gemeinsame Sprache „miteinander unterhalten". System A in Bild 1 könnte ein Mikrocontroller, System B ein Temperatursensor sein. Wenn beide Systeme in ihren Datenblättern eine gemeinsame Schnittstelle ausweisen, dann kann eine Kommunikation zwischen den beiden Systemen stattfinden. Dabei müssen sowohl die elektrischen Daten wie auch die Art der Datenübertragung (Geschwindigkeit, Zeichenkodierung usw.) zusammenpassen.

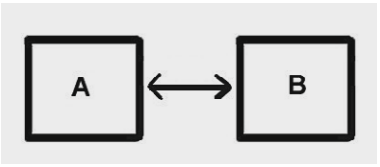


Bild 1: Systeme

Beispiele für digitale Hardwareschnittstellen:

- Parallele Schnittstelle
- Serielle Schnittstellen
  - Asynchrone serielle Schnittstelle (in diesem Artikel beschrieben)
  - Synchrone serielle Schnittstelle
  - SPI
  - I²C
  - 1-Wire
  - USB

Es gibt serielle und parallele Schnittstellen (Bild 2). Bei einer parallelen Datenübertragung (Bild 2 links) werden alle Bits eines Zeichens gleichzeitig über je eine Verbindung übertragen. Diese Art der Übertragung ist sehr schnell, aber es werden viele Drahtverbindungen benötigt. Die wohl bekannteste parallele Schnittstelle war die früher übliche parallele Druckerschnittstelle im Computerbereich.

Alternativ zu dieser parallelen Datenübertragung gibt es Schnittstellen, bei denen die Daten nicht gleichzeitig, sondern nacheinander über eine Verbindungsleitung übertragen werden (Bild 2 rechts). Wie in dem Bild zu erkennen, wird nur eine Verbindungs-

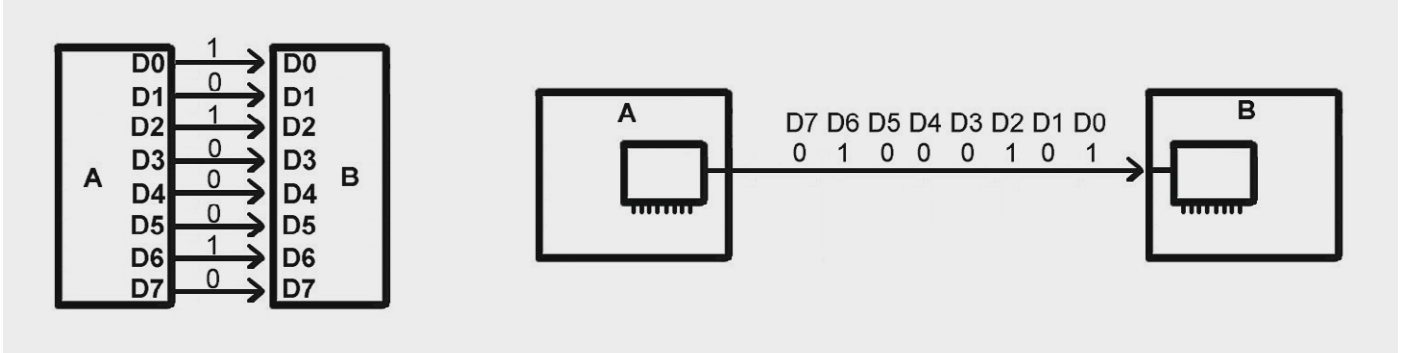


Bild 2: Parallele versus serielle Verbindung

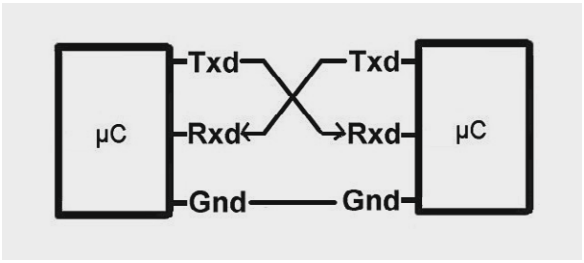


Bild 3: Serielle Verbindung

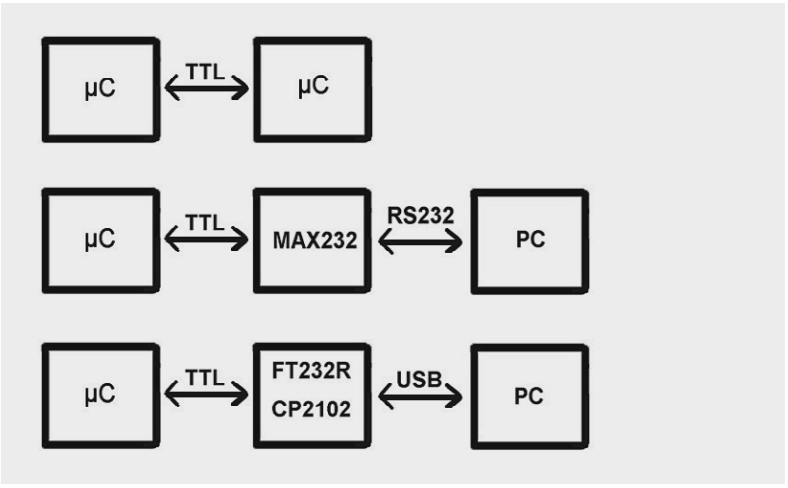


Bild 4: TTL zu TTL (oben), TTL zu RS232 (Mitte) und TTL zu USB (unten)

leitung benötigt, durch die serielle Übertragung der Datenbits wird aber mehr Zeit für die Übertragung aller Bits eines Zeichens benötigt. Da die Systemgeschwindigkeiten immer größer geworden sind, kann man den Geschwindigkeitsnachteil bei serieller Übertragung in Kauf nehmen und dafür Leitungen einsparen. Zusätzlich wird immer noch eine Bezugsleitung (Gnd) benötigt.

Es gibt synchrone und asynchrone serielle Schnittstellen. Bei einer synchronen seriellen Verbindung gibt es ein gemeinsames Taktsignal, welches auf einer zusätzlichen Leitung übertragen wird und die Erkennung der vom Sender gesendeten Bits beim Empfängersystem ermöglicht. Bei einer asynchronen seriellen Verbindung gibt es keine gemeinsame Taktleitung. Die Erkennung der übertragenen Zeichen erfolgt durch gemeinsame Konventionen bzw. Synchronisierungsvereinbarungen.

Ein Zeichen – wie z. B. der Buchstabe E im Wort ELVjournal – wird im Computer oder im Mikrocontroller gemäß der sogenannten ASCII-Tabelle (vgl. Elektronikwissen) durch acht Nullen und Einsen dargestellt. Im sendenden System (A in Bild 2 rechts) werden die acht Bits des Datenbusses üblicherweise durch eine kleine Hardwareeinheit (UART) in eine Abfolge von Nullen und Einsen umgesetzt, welche über die serielle Leitung zu System B (rechts in Bild 2) übertragen und dort wiederum von einer UART-Einheit von seriell zu parallel für den Datenbus „zurückumgesetzt" werden. Man spricht daher auch oft von einer UART-Schnittstelle. Prinzipiell kann das serielle Datenformat statt von einer Hardware-UART-Einheit (UART = Universal Asynchronous Receiver and Transmitter) auch mittels Software erzeugt und auf der Empfängerseite per Software wieder zurückverwandelt werden.

Die serielle Schnittstelle

Hier wird die asynchrone serielle Schnittstelle betrachtet, die meistens einfach nur serielle Schnittstelle genannt wird.

Bei einer seriellen Verbindung gibt es am sendenden System einen Sender-Pin (Txd in Bild 3) und beim Empfängersystem einen Empfangspin (Rxd in Bild 3). Dazu gibt es die Gnd-Leitung für ein gemeinsames Bezugspotential. Wenn Daten auch in die entgegensetzte Richtung übertragen werden sollen, dann wird eine zweite serielle Datenleitung (von rechts nach links in Bild 3) geschaltet, die den Txd-Pin des rechten Systems mit dem Rxd-Pin des linken Pins verbindet.

Bei der Verbindung zweier Systeme ist die elektrische Kompatibilität extrem wichtig. Zwei Systeme (Mikrocontroller oder Sensoren mit Mikrocontroller) mit 5 V können direkt miteinander verbunden werden (TTL, Bild 4 oben). Ebenso funktioniert das bei zwei Systemen mit

Elektronikwissen

ASCII-Tabelle

ASCII steht für „American Standard Code for Information Interchange" und stellt eine Zeichenkodierung dar. In der Computertechnik werden Informationen digital, also als Einsen und Nullen, übertragen. Mehrere dieser atomaren (Bit-)Informationen können zusammengefasst als Zeichen interpretiert werden. Dabei hat sich die Darstellung von Zeichen gemäß des sogenannten ASCII-Zeichensatzes im PC-Bereich durchgesetzt. In einer Tabelle wird dargestellt, welche Bitfolgen als welches Zeichen zu interpretieren sind. Die ASCII-Tabelle definiert die Darstellung von 128 Zeichen. Dabei gibt es sogenannte Steuerzeichen (oder nicht druckbare Zeichen) im Bereich von dezimal 0 bis 31. Von dezimal 32 bis 127 werden die weiteren Zeichen – z. B. alle Buchstaben des Alphabets – dargestellt. In ASCII-Tabellen werden die verschiedenen Zahlendarstellungen (dezimal, binär, hexadezimal) für jedes unterstützte Zeichen aufgeführt.

Beispiele:

Dezimal	Hexadezimal	Binär	Zeichen
32	20	00010 0000	Leerzeichen
65	41	0100 0001	A
...	...	...	...
69	45	0100 0101	E
76	4C	0100 1100	L
86	56	0101 0110	V
...	...	...	...

Die komplette ASCII-Tabelle findet man z. B. unter:
www.ascii-code.com oder
https://blog.nerdmind.de/page/ascii-tabelle/

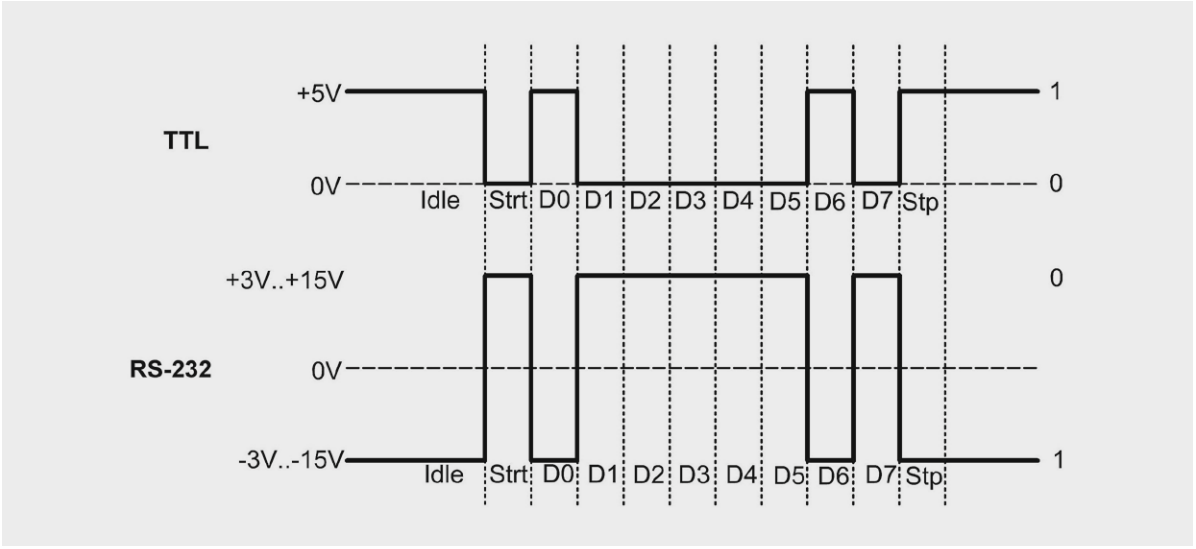


Bild 5: Serielle Datenübertragung (TTL und RS232): Bitübertragung des Buchstabens A

3,3 V. Sobald ein System mit 5 V und das andere System mit 3,3 V betrieben wird, muss man über Pegelwandler nachdenken. Bei der Verbindung von einem Mikrocontroller (oder Sensor mit Mikrocontroller) mit einem PC trifft man auf andere Spannungswerte und muss deshalb geeignete Wandler zu RS232 bzw. USB zwischen die Systeme schalten (Bild 4 Mitte und unten). Es gibt sogar Funkmodule, die die Strecke zwischen den beiden Kommunikationspartnern überbrücken können.

Bild 5 zeigt die Spannungsverhältnisse von 5-V-TTL-Systemen im Vergleich zu einer traditionellen seriellen RS232-Schnittstelle. Bei TTL-Systemen, wie man sie im Mikrocontrollerbereich vorfindet (5 V oder 3,3 V), wird ein High Level oder eine logische 1 durch Spannungswerte nahe der Betriebsspannung dargestellt. Ein Low Level oder eine logische 0 wird durch Spannungswerte nahe 0 V dargestellt (Bild 5 oben). Bei einem System mit RS232-Pegeln, wie es früher bei seriellen PC-Schnittstellen oder heutzutage noch bei manchen Messgeräten der Fall ist, wird eine logische 1 durch Spannungswerte zwischen –3 und –15 V und eine logische 0 durch Spannungswerte zwischen +3 und +15 V dargestellt (Bild 5 unten).

Neben den reinen Spannungsverhältnissen bei der Übertragung kommt es bei der asynchronen seriellen Schnittstelle auf eine gemeinsame Konvention für die Inhalte an. Sender und Empfänger müssen von derselben Übertragungsgeschwindigkeit ausgehen, damit der Empfänger die Spannungswerte auf der Leitung korrekt als logische Signale interpretieren kann. Man spricht hier von der Bitrate als Bits pro Sekunde. Wie in Bild 5 dargestellt, gibt es einen Ruhezustand (Idle), in dem auf der Übertragungsleitung (bei TTL-Systemen) 5 V liegen. Dann gibt es ein Startbit (0 V), welches dem Empfänger den Beginn einer Zeichenübertragung signalisiert. Nach dem Startbit folgen die Datenbits. In Bild 5 sieht man die acht Datenbits D0 bis D7, welche nacheinander übertragen werden. Die Anzahl von acht Datenbits hat sich etabliert. Nach den Datenbits könnte ein Paritätsbit, also ein Prüfbit, mit dem die Korrektheit der übertragenen Datenbits geprüft werden kann, folgen. Die Übertragung eines Zeichens wird schließlich durch ein oder mehrere Stoppbits abgeschlossen.

Für die Übertragung eines Zeichens verwendet man also üblicherweise:

1. ein Startbit,
2. n Datenbits (5 bis 9, meist 8),
3. evtl. ein Paritätsbit (meist N = no Paritybit; manchmal E = Even = gerade Parität oder Odd = ungerade Parität),
4. ein oder zwei Stoppbit(s) (meist 1).

Die Verwendung von einem Startbit, acht Datenbits, keinem Partitätsbit und einem Stoppbit kennzeichnet man zusammenfassend als Übertragungsformat 8N1, wobei die 8 für die Anzahl der Datenbits, das N für no Paritybit und die 1 für ein Stoppbit steht. Ein Zeichen besteht also insgesamt aus zehn Bits (Bild 5). Man kann auch z. B. 7E1 finden, was dann ein Übertragungsformat mit einem Startbit, sieben Datenbits, einer geraden Parität (E = Even) und einem Stoppbit bedeuten würde.

Durch die Zusammenfassung dieser Bits lässt sich ein komplettes Zeichen darstellen (vgl. Elektronikwissen ASCII-Tabelle). Die Anzahl der pro Sekunde übertragenen Zeichen (Symbolrate) hat die Einheit Baud. Historisch gesehen (aus der Zeit der Telegrafie) haben sich bestimmte Baudraten etabliert (2400, 4000, 9600, 14.400, 19.200, 28.800 usw.)

Und nun die wichtigste Regel:
Übertragungsformat (z. B. 8N1) und Symbolrate (z. B. 9600 Baud) müssen bei Sender und Empfänger gleich eingestellt sein, sonst kann die Übertragung nicht funktionieren!

In der Praxis hat man oft bereits ein System bzw. eine Platine – z. B. einen Arduino oder einen Raspberry – und möchte einen Sensor oder einen Aktor über die serielle Schnittstelle ansprechen. Man muss dann im Datenblatt oder in den technischen Details auf der ELV-Produktseite nachlesen, ob der in Betracht gezogene Sensor/Aktor eine serielle Schnittstelle anbietet und welche Parameter (Baud-

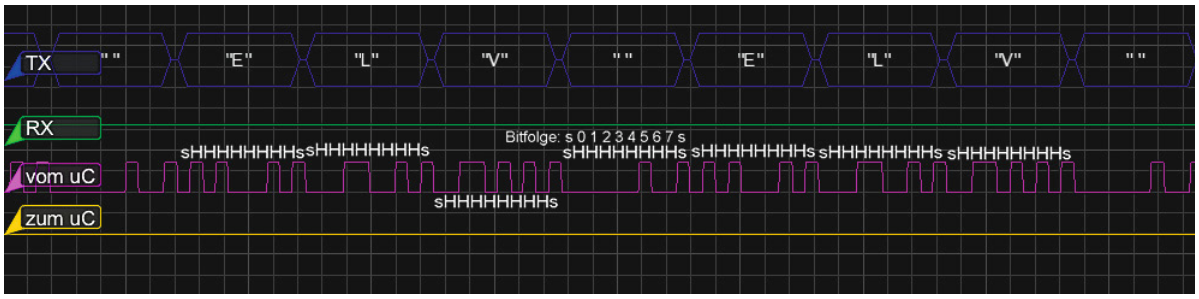


Bild 6: Logikanalyzer

rate und Übertragungsformat) der Sensor anbietet. Bei manchen Geräten sind die Parameter der seriellen Schnittstelle in Grenzen – z. B. durch Jumper – einstellbar, aber bei vielen Sensoren/Aktoren sind die Parameter fest vorgegeben. In jedem Fall muss man dann im eigenen System (im eigenen oder gekauften/heruntergeladenen Programm) die Parameter passend einstellen.

Falls etwas bei der seriellen Verbindung nicht wie erwartet funktioniert, kann man einen Logikanalyzer (z. B. Oscium-Logikanalysator LogiScope, Best.-Nr. CK-11 53 40 oder SmartScope, Best.-Nr. CK-12 37 24) zwischen die serielle Verbindungsleitung und Gnd klemmen und sich den Signalverlauf direkt am Bildschirm ansehen (Bild 6). Dabei sind auch im Logikanalyzer die Parameter entsprechend einzustellen. Die meisten Logikanalyzer sind heutzutage sogar in der Lage, den seriellen Datenstrom zu interpretieren und als Zeichen zu dekodieren. In Bild 6 sieht man als violette Linie die reinen Spannungspegel (zur Illustration wurde die Abgrenzung der Bits an einigen Bit-Paketen ergänzt), und in der oberen, blauen Zeile sieht man die dekodierten Zeichen. Falls kein Logikanalyzer vorhanden ist, kann man serielle Datenströme auch sehr gut mit einem PC und einem sogenannten Terminalprogramm wie z. B. HTerm [1] überprüfen. Auch im Terminalprogramm müssen die Übertragungsparameter richtig eingestellt werden!

Anwendungsbeispiele

Beispiele von Einsatzbereichen sind:

- Verbindungen zwischen Mikrocontrollern (inkl. Arduino, Raspberry u. Ä.)
- Verbindungen zwischen Mikrocontroller und PC (mit USB-Umsetzer oder RS232-Umsetzer)
- Anbindung von „Sensoren":
  - RTC
  - GPS-Modul (NMEA 0183)
  - FS20-/Wetterdaten-Empfänger
  - Temperatur- , Beschleunigungs-, Gas-, Abstands- oder weitere Sensoren
  - Anbindung von Digitalmessgeräten, Fingerabdrucksensoren

Elektronikwissen

Baudraten, Baudratenquarz

Bei der asynchronen seriellen Kommunikation ist es wichtig, dass Sender und Empfänger dieselben Übertragungsparameter (Baudrate, Anzahl Daten-, Paritäts- und Stoppbits) benutzen und – da es keine Taktleitung gibt – die Taktfrequenzen der beteiligten Geräte mit der verwendeten Baudrate zusammenpassen.

Die Synchronisierung erfolgt durch das Startbit. Eine halbe Bitlänge nach Ende des Startbits tastet der Empfänger den Spannungspegel ab, danach jeweils eine Bitlänge später. Wenn die Taktfrequenzen und die Baudrate nicht zusammenpassen, verschiebt sich der Abtastzeitpunkt von der Mitte des empfangenen Bits nach hinten oder nach vorne und es kann zu falschen Werten kommen. Damit es nicht zu den beschriebenen Abtastfehlern kommt, darf man weder einen instabilen internen RC-Oszillator noch einen beliebigen Quarz als Taktgeber für den Mikrocontroller verwenden.

Vielmehr muss man einen Quarz mit zunächst einmal „krumm" aussehender Frequenz verwenden, damit die Übertragung fehlerfrei erfolgen kann. Derartige gut passende Quarze nennt man Baudratenquarze. Übliche Frequenzen von Baudratenquarzen sind 1,8432 MHz, 3,6864 MHz, 7,3728 MHz, 11,0592 MHz, 14,7456 MHz, 18,4320 MHz.

Die zu verwendende Baudrate ist entweder durch einen Kommunikationspartner vorgegeben (z. B. ELV-Modul oder Sensor) oder kann frei gewählt werden, wenn auf keiner Seite etwas vorgegeben ist. Historisch bedingt haben sich Baudraten von 2400, 4800, 9600, 14.400, 19.200, 28.800 usw. etabliert.

Außer dem manuellen Ausrechnen zusammenpassender Taktfrequenz-Baudrate-Paarungen gibt es verschiedene Möglichkeiten, zusammenpassende Werte für Prozessortakt und Baudrate zu ermitteln:

1. Im Datenblatt des verwendeten Mikrocontrollers (beim ATmega88 in Kapitel 20.11) findet man Tabellen mit zusammenpassenden Takt-Baud-Kombinationen
2. In Baudratentabellen bzw. -rechnern im Internet

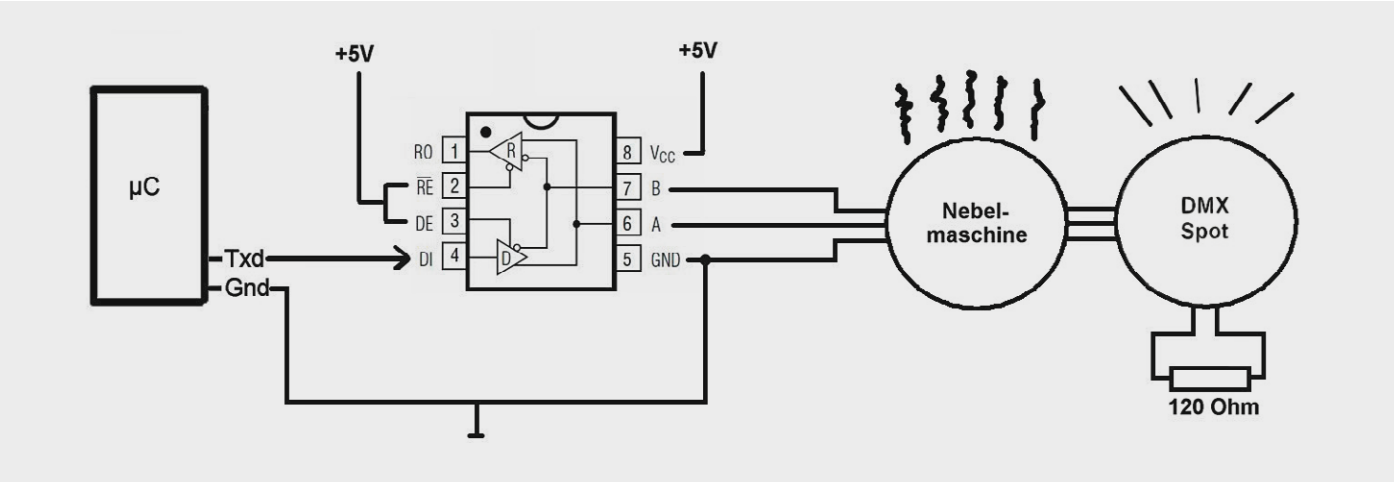


Bild 7: DMX

- Anbindung von „Aktoren":
  - FS20-Sender
  - MP3-Soundmodul
  - DMX-Geräte (Nebelmaschine, LED-Spot)
  - Steuerung von Geräten, Druckern, Robotern
  - Kleine Bildschirme
  - Funk-Sender/Empfänger-Bausteine

Empfohlene Produkte	Best.-Nr.	Preis
Arduino Uno	CK-10 29 70	€ 27,95
Mini-USB-Modul UM2102, Komplettbausatz	CK-09 18 59	€ 5,95
myAVR mySmartUSB MK2	CK-07 28 25	€ 27,95
ELV-Highspeed-Mini-USB-Modul UM-FT2232H, Komplettbausatz	CK-09 93 47	€ 16,95
ELV FS20-UART-Sender FS20 US, Komplettbausatz	CK-09 87 89	€ 19,95
ELV FS20- und Wetterdaten-UART-Empfänger FS20 WUE, Komplettbausatz	CK-10 38 66	€ 14,95
Real-Time-Clock-DCF-Modul mit I²C, SPI und UART-Schnittstelle, RTC-DCF, Komplettbausatz	CK-13 05 41	€ 12,95
MP3-Soundmodul MSM3, Komplettbausatz	CK-10 57 29	€ 39,95
ELV TTL-nach-RS232-Umsetzer, Komplettbausatz	CK-03 84 39	€ 6,95
RS232-Shield, serielle Schnittstelle für pcDuino/Arduino	CK-11 89 81	€ 19,95
Linker-Kit-RS485-Schnittstelle für Raspberry Pi	CK-11 87 24	€ 12,95
Ethernet-UART-Gateway EUG 100, Komplettbausatz	CK-08 52 59	€ 39,95
Ethernet-UART-Gateway EUG 100	CK-09 20 31	€ 34,95
Oscium-Logikanalysator LogiScope	CK-11 53 40	€ 159,95
LabNation SmartScope 2-Kanal-USB-Speicher-Oszilloskop	CK-12 37 24	€ 229,–
Velleman-Nebelmaschine, DMX-Steuerung, 1500 W	CK-11 85 95	€ 129,95

Zum Thema DMX zeigt Bild 7 ein kleines Beispielszenario, in dem ein Mikrocontroller benutzt wird, um einen DMX-konformen seriellen Datenstrom auszugeben und dadurch (nach Zwischenschalten eines RS485-Pegelwandler-ICs) viele DMX-Geräte – LED-Spots, Nebelmaschinen usw. – anzusteuern.

Fazit

Seit den 1960er-Jahren hat sich die serielle Schnittstelle (genauer die asynchrone serielle Schnittstelle) als einfacher und robuster Standard (kommend aus dem Bereich der Fernschreiber, Modems und Terminalanbindungen) zur Verbindung unterschiedlicher Systeme bewährt. Bis zum Erscheinen von USB hatte jeder PC mindestens eine serielle Schnittstelle. Auch heute ist die serielle Verbindung im Hobby- und im Profi-Bereich (in zum Teil leichten Variationen, die hauptsächlich die elektrische Übertragung betreffen) verbreitet. Die serielle Schnittstelle (im engeren Sinne) ist weit verbreitet, einfach verständlich und robust.

Im nächsten ELVjournal wird die SPI-Schnittstelle erklärt, welche für viele Sensoren/Aktoren verwendet wird. ELV

Weitere Infos:

[1] Windows-Terminalprogramm HTerm: www.der-hammer.info/terminal/

Preisstellung Oktober 2016 – aktuelle Preise im Web-Shop