30 Die serielle Schnittstelle

Neben der parallelen Schnittstelle ist im PC nach wie vor die serielle Schnittstelle aufgrund ihrer Vielseitigkeit von großer Bedeutung. Hier können Sie so unterschiedliche Geräte wie z.B. eine Maus, ein Modem, einen Plotter und natürlich auch einen geeigneten Drucker anschließen. Aufbau, Funktionsweise und Programmierung der seriellen Schnittstelle sind Gegenstand dieses Kapitels.

30.1 Serielle Datenübertragung

Die parallele Datenübertragung haben Sie bereits in Kapitel 29 kennen gelernt. Kennzeichen der parallelen Datenübertragung ist, dass die Bits eines Datenbyte und gegebenenfalls ein Paritätsbit parallel, d.h. gleichzeitig über eine Mehrzahl von Datenleitungen übergeben werden. Für ein Datenbyte und ein Paritätsbit sind also neun Datenleitungen notwendig. Bei einer seriellen Übertragung werden die einzelnen Datenbits und das möglicherweise vorhandene Paritätsbit dagegen über eine einzige Datenleitung in zeitlicher Folge, d.h. seriell, übergeben. Man könnte nun einwenden, dass die parallele Schnittstelle beim Ausdruck eines größeren Dokuments die Datenbytes ja auch zeitlich nacheinander, also seriell, übergibt. Der Unterschied besteht jedoch darin, dass bei der parallelen Übertragung die einzelnen Dateneinheiten – das sind die Datenbytes – als Ganzes übergeben werden, während bei der seriellen Übertragung auch die Dateneinheiten in einzelne Bits unterteilt und bitweise übergeben werden.

30.1.1 Synchrone und asynchrone Übertragung

Bei der seriellen Datenübertragung unterscheidet man noch zwischen synchronem und asynchronem Datenaustausch. Bei einer synchronen Übertragung werden über eine getrennte Leitung noch ein oder mehrere Signale übertragen, die angeben, wann das nächste Bit auf der Datenleitung gültig ist. Diese Signale können durch Taktsignale einer Taktquelle oder durch Handshake-Signale der Art *Request* und *Acknowledge* gebildet werden. Der große Vorteil der synchronen Übertragung ist, dass der Empfänger auf unterschiedliche Taktraten reagieren kann, solange seine Maximalfrequenz nicht überschritten wird, indem er einfach z.B. den Übergang low-high des Taktsignals erfasst.

Im Gegensatz dazu ist bei asynchroner Übertragung ein Minimum an Synchronisationsinformation in die Datenbits selbst eingebettet, wobei Sender und Empfänger mit derselben Taktfrequenz arbeiten müssen. Die eingebettete Synchronisationsinformation besteht aus einem so genannten Startbit, das den Beginn einer Dateneinheit anzeigt, und mindestens einem Stoppbit, das das Ende der Dateneinheit angibt. Nimmt man die Paritätsinformation hinzu, die ja häufig auch bei paralleler Datenübertragung verwendet wird, besteht eine serielle Dateneinheit oder SDU (*Serial Data Unit*) aus einem Startbit, den Datenbits, gegebenenfalls einem Paritätsbit und mindestens einem Stoppbit. Gegenüber der synchronen seriellen Datenübertragung ergibt sich hier also ein Overhead durch Start- und Stoppbits.

30.1.2 Parität und Baudrate

Die Parität ist ein einfacher und leider auch wenig leistungsfähiger Schutz gegen Übertragungsfehler. Durch die Parität können nur einfache Bitfehler zuverlässig erkannt werden, Bündelfehler mit mehreren gestörten Bits werden mit 50-prozentiger Wahrscheinlichkeit nicht erfasst. Die Pari-

tät eignet sich also nur für kurze und wenig störanfällige Übertragungswege. Für andere Anwendungen sind CRC-Codes wesentlich zuverlässiger, aber auch komplizierter zu berechnen. Der Vorteil der Parität ist aber, dass nahezu alle seriellen Schnittstellenbausteine die Erzeugung und Prüfung von Paritätsbits hardware-mäßig unterstützen. Insgesamt unterscheidet man fünf verschiedene Paritäten:

- **keine Parität:** Es wird kein Paritätsbit eingefügt.
- **gerade Parität:** Das Paritätsbit wird so gesetzt, dass in den Datenbits und dem Paritätsbit zusammen eine gerade Anzahl von Einsen vorkommt.
- **ungerade Parität:** Das Paritätsbit wird so gesetzt, dass in den Datenbits und dem Paritätsbit zusammen eine ungerade Anzahl von Einsen vorkommt.
- **Mark:** Das Paritätsbit wird stets auf den Wert »1« gesetzt.
- **Space:** Das Paritätsbit wird stets auf den Wert »0« gesetzt.

An dieser Stelle ein paar Bemerkungen zu den reichlich wertlosen Paritäten Mark und Space. Mark und Space eignen sich tatsächlich nur dazu, Fehler im Paritätsbit selbst zu erfassen; wenn eine SDU mit gelöschtem oder gesetztem Paritätsbit einläuft und eigentlich Mark- bzw. Space-Parität vorhanden sein sollte, liegt ein Fehler im Paritätsbit vor. Über die Richtigkeit der Datenbits sagt das natürlich überhaupt nichts aus. Umgekehrt können die Datenbits zerstört sein, der Empfänger merkt aber nichts davon, wenn das Paritätsbit unbeschädigt ist. Mark und Space sind also nur von sehr geringem bis umständlichem Wert.

Eine weitere Größe im Zusammenhang mit der seriellen Datenübertragung, die häufig Anlass zu Missverständnissen gibt, ist die *Baudrate*. Benannt nach dem französischen Mathematiker J.M.E. Baudot, gibt sie lediglich die Anzahl der Signaländerungen eines Übertragungskanals je Sekunde an. Da bei den üblichen seriellen Schnittstellen die Signaländerungen zeitlich äquidistant sind und eine sehr einfache binäre Datenkodierung ausgeführt wird – ein logisch hoher Pegel gleich Mark entspricht einer »1«, ein logisch niedriger Pegel gleich Space entspricht einer »0« –, ist die Baudrate hier gleich der Anzahl der übertragenen *Bit pro Sekunde* (bps), wenn man auch Start-, Paritäts- und Stoppbits einschließt. Bei leistungsfähigen Kodierungen mit Datenkompressionsverfahren, wie sie etwa bei Modems zum Einsatz kommen, werden Datenraten (in bps) erzielt, die deutlich über der Baudrate liegen.

Auszug aus der Klasse gnu.io.SerialPortInterface

public static final int	PARITY_NONE	0
public static final int	PARITY_ODD	1
public static final int	PARITY_EVEN	2
public static final int	PARITY_MARK	3
public static final int	PARITY_SPACE	4
public static final int	STOPBITS_1	1
public static final int	STOPBITS_1_5	3
public static final int	STOPBITS_2	2

p - the parity schema: PARITY_NONE, PARITY_ODD, PARITY_EVEN, PARITY_MARK or PARITY_SPACE