24.1.5.1 RS232

Viele Computer verfügen heute kaum noch über eine echte serielle Schnittstelle mit einem RS232-Stecker wie zum Beispiel 'Sub-D-9-polig'. Ersatz bieten in diesem Fall USB-RS232-Adapter, die eine Datenübertragung mit Peripherie-Geräten mit serieller Schnittstelle über einen USB-Port Ihres Computers ermöglichen. So ein USB-RS232-Adapter bietet Ihnen zum Beispiel (Abbildung 24.1.5.1.1) eine Schnittstelle mit 1×RS232-Stecker (Sub-D-9-polig) und 1×USB-A-Stecker. Im RS232-Stecker ist ein Hardware-Chip (USB UART Chip als USB2ToSerial-Bridge) eingebaut, der u.a. auch die notwendige USB-Seriell-Umwandlung vornimmt. Linux Treiber sind in allen modernen Distributionen verfügbar. Der passende Treiber erzeugt im Betriebssystem einen virtuellen seriellen Port, über den die Kommunikation mit dem USB-RS232-Adapter erfolgt. Der Adapter verhält sich gegenüber Software wie eine echte serielle RS232-Schnittstelle und wird auch als solche erkannt.

Abbildung 24.1.5.1.1: USB-RS232-Adapter mit Verlängerungskabel

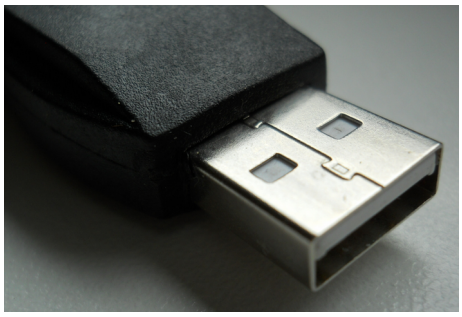

Abbildung 24.1.5.1.2: USB-RS232-Adapter (USB)

Abbildung 24.1.5.1.3: USB-RS232-Adapter (RS232)

24.1.5.1.1 Serielle Schnittstellen

Wenn Sie keine oder nur geringe Erfahrungen im Umgang mit seriellen Schnittstellen besitzen, dann kann Ihnen ein Blick auf den Inhalt der Webseiten unter http://tldp.org/HOWTO/Serial-HOWTO.html empfohlen werden.

Mit den folgenden Befehlen in einer Konsole können Sie erkunden, ob der zu untersuchende Computer über serielle Schnittstellen verfügt.

Eine Liste der erkannten seriellen I/O-Geräte erhalten Sie mit den folgenden Befehlen:

```
hans@mint-183 ~ $ dmesg | grep ttyS
hans@mint-183 ~ $ dmesg | grep ttyUSB
```

Fazit bei der Überprüfung des Laptops des Autors (Baujahr 2017): Weit und breit ist keine serielle Schnittstelle zu entdecken – beide Ausgaben sind leer.

Nach dem Anstecken von zwei USB-RS232-Adaptern am Computer ergab sich die folgende Anzeige, da jetzt nur nach USB-RS232-Adaptern gesucht wurde:

```
hans@mint-183 ~ $ ls -l /dev | grep serial
drwxr-xr-x  4 root root  80 Apr 20 18:14 serial
```

Wenn die Ausgabe nicht leer ist und der Ordner /dev/serial existiert, dann gibt es mindestens einen seriellen USB-RS232-Adapter am Computer. Das wird durch die folgende Anzeige der Geräte bestätigt:

```
hans@mint-183 ~ $ ls -l /dev/ttyUSB*
crw-rw---- 1 root dialout 188, 0 Apr 20 14:30 /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 1 Apr 20 14:31 /dev/ttyUSB1
```

Hinweise

- Die Gerätedateien /dev/ttyUSB0 und /dev/ttyUSB1 werden nur dann im Ordner */dev* temporär angelegt, wenn ein solches I/O-Gerät als USB-RS232-Adapter angeschlossen ist.
- Der Gerätename steht am Ende der Zeile. Das erleichtert das Parsen der einzelnen Zeilen, wenn Sie den Befehl in einer Shell- oder EXEC-Instruktion in Ihren RS232-Projekten verwenden.
- Die Bezeichner ttyUSB0 und ttyUSB1 sind die Gerätenamen, die den seriellen Ports von dem USB2Serial-Treiber vergeben wurden.

Von der Verwendung des dmesg-Befehls ist dann abzuraten, wenn Sie den USB-RS232-Adapter mehrfach einstecken und wieder abziehen, weil die redundanten Ausgaben sehr umfangreich werden können:

```
hans@mint-183 ~ $ dmesg | grep ttyUSB
[   97.789809] usb 3-1: pl2303 converter now attached to ttyUSB0
[  129.010906] usb 2-1.3: pl2303 converter now attached to ttyUSB1
...
[ 8186.990521] pl2303 ttyUSB0: pl2303 converter now disconnected from ttyUSB0
[ 8189.033382] pl2303 ttyUSB1: pl2303 converter now disconnected from ttyUSB1
[ 9009.588711] usb 2-1.3: pl2303 converter now attached to ttyUSB0
[ 9015.188868] usb 3-1: pl2303 converter now attached to ttyUSB1
[10652.765735] pl2303 ttyUSB0: pl2303 converter now disconnected from ttyUSB0
hans@mint-183 ~ $
```

Entscheidend sind bei zwei USB-RS232-Adaptern die letzten Zeilen. Sie müssen beim Parsen der Ausgabe erkennen, ob beide USB-RS232-Adaptern eingesteckt sind oder beide entfernt wurden oder nur ein USB-RS232-Adapter eingesteckt ist.

Eine typische Ausgabe von dmesg bei echten seriellen Schnittstellen sieht so aus:

```
hans@mint-183 ~ $  dmesg | grep ttyS | grep 00:
[    0.962453] 00:02: ttyS0 at I/O 0x3f8 (irq = 4, base_baud = 115200) is a 16550A
```

24.1.5.1.2 Zugriff auf eine serielle Schnittstelle

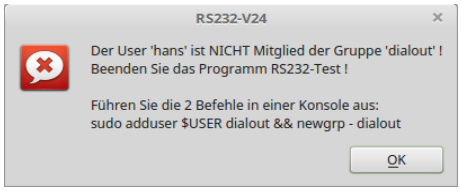

Abbildung 24.1.5.1.4: Fehlermeldung

```
hans@mint-183 ~ $ ls -l /dev/ttyUSB*
crw-rw---- 1 root dialout 188, 0 Apr 20 14:30 /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 1 Apr 20 14:31 /dev/ttyUSB1
```

Wie Sie erkennen, haben standardmäßig nur der Benutzer root und die Mitglieder der Gruppe 'dialout' Zugriff auf die beiden erkannten seriellen Schnittstellen (USB-RS232-Adapter). Deshalb ist es erforderlich, dass der Benutzer 'hans' Mitglied der Gruppe 'dialout' wird.

Temporäre Aktivierung eines Users

Mit diesem Befehl wird der aktuelle Benutzer 'hans' aus der Gruppe 'hans' für die *aktuelle* Sitzung Eigentümer für das Device /dev/ttyUSB0:

```
hans@linux:~$ sudo chown -R hans:hans /dev/ttyUSB0
[sudo] password for hans:
hans@linux:~$
```

Der Befehl 'sudo adduser username gruppenname' fügt den angegebenen – bereits vorhandenen Benutzer – permanent der angegebenen Gruppe hinzu.

Zuerst wird der Benutzer 'hans' zur Gruppe 'dialout' hinzugefügt:

```
hans@mint-183 ~ $ sudo adduser hans dialout
[sudo] Passwort für hans:
Füge Benutzer »hans« der Gruppe »dialout« hinzu ...
Benutzer hans wird zur Gruppe dialout hinzugefügt.
Fertig.
```

Dann wird die Gruppenzugehörigkeit aktiviert – was ohne Rückmeldung erfolgt, wenn das Aktivieren erfolgreich war. Achten Sie auf den Schalter -, der die sofortige Aktivierung auslöst!

```
hans@mint-183 ~ $ newgrp - dialout
hans@linux:~$
```

Abschließend erfolgt zur Kontrolle die Anzeige der Gruppen, zu denen der Benutzer 'hans' gehört:

```
hans@mint-183 ~ $ groups hans
hans : hans adm dialout cdrom sudo dip www-data plugdev lpadmin sambashare
```

Für die Verwendung der seriellen Schnittstellen am Raspberry Pi enthält die Seite http://www.netzmafia.de/skripten/hardware/RasPi/RasPi_Serial.html viele wertvolle Hinweise.

24.1.5.1.3 Datenübertragung zwischen Computern und RS232-Peripheriegeräten

Serielle Schnittstellen werden zur Datenübertragung zwischen Computern und Peripheriegeräten eingesetzt. Die einzelnen Bytes im Datenstrom werden nach einer byteparallelen-bitseriellen Wandlung Byte für Byte in einem seriellen Bit-Datenstrom vom Computer über eine Verbindungsleitung zu den seriellen Peripheriegeräten übertragen. Die Daten von den Peripheriegeräten werden in der seriellen Schnittstelle einer bitseriellen-byteparallelen Wandlung unterzogen. Die Übertragungszeit eines einzelnen Bits ist abhängig von der eingestellten Baudrate. Die Zeitdauer für die Übertragung eines Bits ist der Kehrwert der Baudrate.

Bei einer RS-232-Schnittstelle erfolgt die Datenübertragung asynchron. Das bedeutet, dass die Pause zwischen zwei Bytes beliebig lang sein kann. Deshalb synchronisieren sich Sender und Empfänger für jedes einzelne zu übertragene Zeichen neu. Die Synchronisation erfolgt über ein sogenanntes Start-Bit und die Stopp-Bits, von denen es bei den Stopp-Bits mindestens eines gibt. Während der Pause liegt der Signal-Pegel auf logisch 1.

Soll ein Zeichen übertragen werden, so markiert ein Pegelwechsel von logisch 1 auf logisch 0 durch das Setzen des Start-Bits den Beginn der Übertragung des Datenwortes. Danach werden die einzelnen Bits eines Datenwortes mit dem konstanten Takt (1/Baudrate) übertragen, wobei zuerst das niederwertigste Bit übertragen wird. Ein Datenwort besteht aus 5 bis 8 Bits, wobei 7 oder 8 Bits lange ASCII-kodierte Datenworte als Nutzdaten übliche Wert sind. Abschließend folgt nach einem optionalen Paritätsbit mindestens ein Stopp-Bit – der Signal-Pegel wird auf logisch 1 gesetzt. Die Pause beginnt und der Empfänger wartet auf das nächste Start-Bit ... .

24.1.5.1.4 Hinweise zu den Übertragungsparametern

Übertragungsparameter Baudrate – SerialPort.Speed:

Die Gambas-Dokumentation gibt für die Eigenschaft *SerialPort.Speed* den Hinweis, dass die Baudrate ein gültiger Standardwert sein muss, der vom Treiber der seriellen Schnittstelle unterstützt wird. Wenn Sie zum Beispiel den Typ des UART (Universal Asynchronous Receiver/Transmitter) in den beiden erkannten USB-RS232-Adaptern kennen würden, dann wären auch Aussagen über die maximal unterstützte Baudrate möglich:

```
hans@mint-183 ~ $ setserial -g /dev/ttyUSB[01]
/dev/ttyUSB0, UART: 16654, Port: 0x0000, IRQ: 0
/dev/ttyUSB1, UART: 16654, Port: 0x0000, IRQ: 0
```

In einem Datenblatt für den 16654-UART stand: Serielle Ports, die von einem 16654-UART gesteuert werden, erlauben Baudraten bis zu 115200 Baud.

Bis zur stabilen Gambas-Version 3.12.2 erlaubte die Klasse gb.net lediglich die Einstellung der Baudraten 50, 75, 110, 150, 200, 300, 600, 1200, 2400, 4800, 9600, 19200 und 38400 Baud. Höhere Baudraten wurden nicht getestet. Für alle späteren Gambas-Stable-Version sind beliebige Baudraten erlaubt.

Übertragungsparameter DatenBits – SerialPort.DataBits

Sie können über den Parameter SerialPort.DataBits festlegen, aus wieviel DatenBits ein Datenwort besteht. Verwenden Sie eine der folgenden Konstanten: SerialPort.Bits5, SerialPort.Bits6, SerialPort.-Bits7 oder SerialPort.Bits8. Die Einstellung SerialPort.DataBits = SerialPort.Bits8 ist üblich.

Übertragungsparameter Paritätsprüfung – SerialPort.Parity

Eine Paritätsprüfung ermöglicht die Prüfung auf 1-Bit-Fehler bei der Übertragung eines Datenwortes. Bei der Paritätsprüfung, sofern sie als gerade oder ungerade eingeschaltet wurde, wird auf der Senderseite zuerst die Anzahl der 1-Wert-Bits im Datenwort ermittelt. Ist die ermittelte Anzahl ungerade, dann wird zum Beispiel bei gerader Parität das Paritätsbit auf 1 gesetzt, damit die Anzahl der 1-Wert-Bits gerade ist. Das sind die Einstellmöglichkeiten für diesen Übertragungsparameter: Even parity (E), No parity (N), Odd parity: (O) oder SerialPort.Parity = SerialPort.None, SerialPort.Parity = SerialPort.Even, SerialPort.Parity = SerialPort.Odd. Da auf der Empfängerseite mit gleichen Übertragungsparametern gearbeitet werden muss, können dort durch Vergleich nur 1-Bit-Datenfehler erkannt werden. Aus diesem Grund wird häufig auf eine Paritätsprüfung verzichtet und der Übertragungsparameter auf *No parity* (N) gesetzt.

Übertragungsparameter StoppBits – SerialPort.StopBits

Im Normalfall können Sie 1 Stopp-Bit (SerialPort.Stopbits = Serialport.SerialPort.Bits1) oder 2 Stopp-Bits (SerialPort.Stopbits = Serialport.SerialPort.Bits2) setzen. Beachten Sie, dass bei einem 5-Bit-Datenwort das Setzen von 2 Stopp-Bits zu 1.5 Stopp-Bits führt, wie Pegel-Zeit-Messungen bestätigten. Es gibt in der Klasse SerialPort keine Eigenschaft SerialPort.Startbit, denn es gibt nur ein Start-Bit mit festem Signal-Pegel (logisch 0) und das wird automatisch gesetzt.

24.1.5.1.5 Hinweise zur Datenfluss-Steuerung

Eine Steuerung des Datenflusses wird nur für den Fall benötigt, dass ein Gerät – wie zum Beispiel ein Computer – Daten schneller liefern kann, als ein serielles Peripheriegerät sie verarbeiten kann. In diesem Fall muss das Peripheriegerät dem Computer signalisieren, dass es aktuell keine weiteren Daten über die RS232-Schnittstelle empfangen kann. Die Signalisierung kann entweder über zwei spezielle Steuerzeichen im Datenstrom (Software-Handshake) oder über das Setzen und Auslesen der Pegel bestimmter Steuerleitungen vorgenommen werden (Hardware-Handshake).

Software-Handshake: Es werden das Steuerzeichen Xoff (DC3, 13_{hex}, 19_{dec}) oder das Steuerzeichen Xon (DC1, 11_{hex}, 17_{dec}) im Datenstrom eingesetzt, um den Status der beiden seriellen Geräte zu erkennen. Die beiden Steuerzeichen dürfen verständlicherweise *nicht* im zu übertragenen Nutzdatenpaket enthalten sein!

Hardware-Handshake: Die DTR-Steuerleitung (out) signalisiert, dass das Datenendgerät (hier der PC) bereit ist. Die DSR-Leitung (in) signalisiert dem Datenendgerät, dass die Gegenstelle bereit ist. Die beiden Steuerleitungen RTS (out) und CTS (in) werden für die Datenflusssteuerung verwendet.

Verwenden Sie eine der folgenden Konstanten: SerialPort.Hardware, SerialPort.Software oder auch SerialPort.Both, um die Art der Datenflusssteuerung festzulegen. Wird keine Datenflusssteuerung benötigt, setzen Sie die Eigenschaft SerialPort.FlowControl auf SerialPort.None.

24.1.5.1.6 Initialisierung SerialPort-Objekt

Wenn Sie eine RS232-Schnittstelle öffnen, die Sie entweder in der Gambas-IDE aus der Werkzeugsammlung im Reiter 'Netzwerk' auf das Formular gezogen haben oder im Quelltext zum Beispiel mit *hRS232 = New SerialPort As "hRS232"* erzeugen, wird die RS232-Schnittstelle mit den folgenden Übertragungsparametern initialisiert:

Abbildung 24.1.5.1.5: Standard-Übertragungsparameter für einen (neuen) seriellen Port (Gambas-IDE)

Änderungen der Übertragungsparameter können Sie im Objekt-Inspektor oder im Quelltext vornehmen. Verwenden Sie dann konsequent die vorgegebenen Konstanten der Klasse SerialPort (gb.net):

```
...
hRS232.Parity = SerialPort.None
hRS232.StopBits = SerialPort.Bits1
hRS232.FlowControl = SerialPort.None
...
```