Serielle Verbindung

(Version 1.0 September 2015)

www.b-und-l-service.de
info@www.b-und-l-service.de
J.Beushausen@gmx.de

1 Motivation

Immer wieder tauchen Fragen über das Zusammenspiel zwischen Mikro-Kontrollern und PC-Programmen auf. Meistens geht es dabei um eine serielle Verbindung zwischen beiden Geräten. Die Herausforderung bei der Realisierung liegt dabei in der Beherrschung des Zusammenspiels zwischen den drei Komponenten Hardware, Kontrollerprogrammierung und PC Programmierung. Im folgenden Text soll eine solche Verbindung beispielhaft vorgestellt werden.

2 Grundsätzliches

Für das PC-Programm kommt Delphi zum Einsatz, das Kontroller-Programm ist mit BASCOM für einen Prozessor der AT-Mega Familie erstellt. Damit sind zwei populäre Entwicklungsumgebungen im Einsatz. Das Protokoll zur Übertragung ist möglichst einfach gehalten. Zur Veranschaulichung werden zunächst sowohl der Client (PC-Programm) wie auch der Server (später auf dem Kontroller in BASCOM) in Delphi auf dem PC entwickelt. Natürlich ließe sich der Server auf einem PC mit den verfügbaren Resourcen und Hilfsmitteln eleganter programmieren, er soll aber hier nur als Beispiel dienen und berücksichtigt dabei schon die eingeschränkten Möglichkeiten des Kontrollers und des BASIC-Dialekts.

3 Verbindung herstellen

Wesentlich für die Auswahl eines physikalischen COM-Ports ist seine Port-Nummer. Unter Windows lautet die Notation COM1 ... COMxx. Der Gerätemanager gibt Auskunft über die im PC verfügbaren Schnittstellen. Außer den (heute leider kaum noch) auf dem Mother-Board integrierten „echten“ COM-Ports (meist COM1 und Eventuell COM2) können dies auch USB zu Seriell Wandler sein.

Bevor überhaupt ein Programm zum Einsatz kommt, muss zunächst eine physikalische Verbindung zwischen den beiden Geräten hergestellt werden.

Kontroller arbeiten meistens mit 5V (ev. sogar mit noch kleineren Spannungen, z.B. 3,3 V). Ihre eingebaute serielle Schnittstelle liefert also auch nur Spannungen von ca. 0 V bis ca. 5V. Eine (normgerechte) serielle Verbindung geht aber von Spannungen zwischen ca. –10V und +10 V aus. Zur Anpassung gibt es spezielle Schnittstellen ICs, sog. Pegelwandler. Es wird hier davon ausgegangen, dass die zu verbindenden Geräte (also PC und PC oder PC und Kontroller) beide mit den erforderlichen Pegeln von ca. +/- 10 V arbeiten. Die serielle Schnittstelle und ihre Festlegungen werden unter https://de.wikipedia.org/wiki/RS-232 ausführlich dargestellt

Im vorliegenden Beispiel kommt eine Verbindung mit nur drei Drähten zum Einsatz.

TX kennzeichnet dabei eine sendende Leitung, RX kennzeichnet eine empfangende Leitung und GND steht für die gemeinsame Masseleitung. Eine serielle Schnittstelle weist üblicherweise noch weitere Anschlüsse auf. Sie dienen im Wesentlichen der Steuerung des Datenflusses. Da diese Art der Steuerung praktisch nicht von Mikrokontrollern verwendet wird, werden diese Anschlüsse auch nicht weiter betrachtet..

Die physikalische Verbindung muss nun wie folgt hergestellt werden.

Gerät 1		Gerät 2
(3) TX1	----------	(2) RX2
(2) RX1	----------	(3) TX2
(5) GND1	----------	(5) GND2

Die Angeben in Klammern sind die Pin-Nummern an einem 9 poligen SUB-D Steckverbinder Neben den o.g. Spannungspegeln müssen noch verschiedene andere Einstellungen übereinstimmen, damit eine korrekte Kommunikation erfolgen kann.

Baudrate	9600 Baud
Datenbits	8
Parität	keine
Stoppbit	1
Flusssteuerung	keine

Dies sind die Einstellungen die z.B. für ein Terminalprogramm, (Hyperterminal, Tera Term) vorgenommen werden müssen.

Einern Überblick zu Details der verschiedenen Schnittstellen findet man unter
http://www.pci-card.com/schnittstellen.html

4 Übertragungsprotokoll

Unter Protokoll wird die Festlegung bestimmter Eigenschaften beim Austausch von Daten verstanden. Es gibt eine Vielzahl von seriellen Protokollen.

4.1 Vorteile eines ASCII Protokolls

Bei der Entwicklung ist es von großem Vorteil auf erprobte Programme (Terminal-Programm) zurückgreifen zu können. Daher soll hier ein ASCII-Protokoll zum Einsatz kommen. Bei der Verwendung eines ASCII-Protokolls können Fehler damit durch „draufschauen“ gefunden werden.

4.2 Protokolldefinition

Die Praxis verlangt oft die Parametrierung eines Prozesses, der auf einem Mikrokontroller läuft. Dies soll eindeutig von einer Datenübertragung von oder zum Kontroller unterschieden werden. Parameter sind in dieser Betrachtungsweise einzelne Werte die spezifische Aufgaben erfüllen, also z.B. Messrate, Grenzwerte oder Umrechnungsfaktoren.

Die Anweisungen an den Kontroller werden hier als Kommandos bezeichnet. Dem Kontroller müssen in einem Kommando also zwei Dinge mitgeteilt werden, erstens um welchen Parameter es sich handelt und zweitens der eigentliche Wert. Weiterhin ist es sinnvoll diese Parameter auch rücklesbar zu gestalten. Das Protokoll soll Kommandos mit und ohne Parameter gestatten.

Als einfache Möglichkeit für die Übertragung von Parametern hat sich folgender Ansatz bewährt:

zum Kontroller senden : Groß-Buchstabe;Wert CR LF

vom Kontroller lesen : Klein-Buchstabe CR LF

Die Abkürzung CR LF steht hier für die Steuerzeichen carriage return und linefeed (ASCII Zeichen 13 und 10)

Bei kleineren Aufgaben können dann die Buchstaben sinnvoll gewählt werden, also z.B. T für Temperatur; U für Spannung und I für Strom. Es ist klar dass diese Art der Namensgebung irgendwann zu Konflikten führt, andererseits können wichtige Parameter-Namen gut im Kopf behalten werden, was wiederum die Arbeit mit einem Terminalprogramm erleichtert.

Neben den Buchstaben enthält der ASCII-Zeichensatz noch ausreichend andere Zeichen, die mit Sonderfunktionen ohne Parameter belegt werden können. Zum Beispiel „?“ für die Abfrage der Firmwareversion und „!“ für einen Programm-Reset.

Zur Rückmeldung kommen die Zeichenfolgen „ok“ und „error“ zum Einsatz. Eine Rückmeldung erfolgt nur für diejenigen Übertragungen, die nicht selbst eine Antwort generieren.

5 Simulation des Mikrokontroller-Programms auf dem PC

Die serielle Schnittstelle wird über das Betriebssystem mit dem API Aufruf CreateFile mit diversen Parametern angesprochen. Da hier sehr viel Detailarbeit erforderlich ist, haben sich dankenswerter Weise andere Menschen die Mühe gemacht entsprechende Komponenten zu entwickeln. Für Delphi gibt es mehrere Komponenten zur Kommunikation über den COM-Port. Beispielhaft seien hier zwei Quellen genannt.

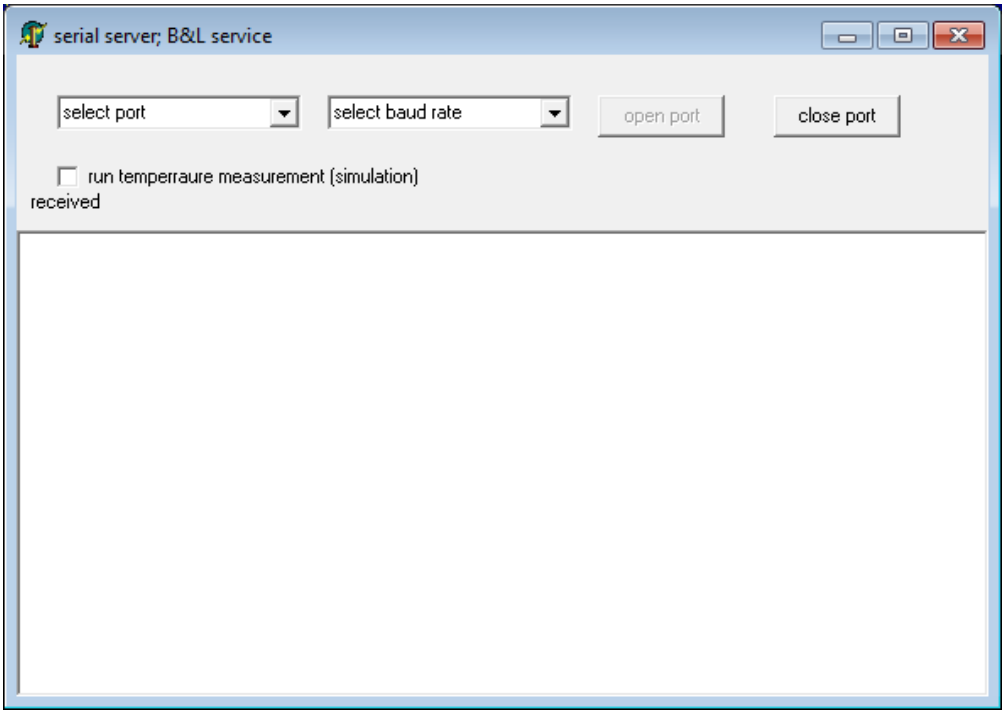
AsyncPro findet sich unter http://sourceforge.net/projects/tpapro/ zum download. Hierbei handelt es sich um eine sehr mächtige Sammlung diverser Komponenten.

TComPort ist deutlich schlanker. Unter http://sourceforge.net/projects/comport/ steht es als download zur Verfügung.

Beide Komponenten müssen vor ihrer Verwendung installiert werden. Sie erleichtern den Umgang mit dem COM-Port außerordentlich und stllen sehr komfortable Möglichkeiten zur Verfügung.

5.1 Server Simulation verwenden

Ohne tatsächlich einen Mikrokontroller zur Verfügung zu haben, kann die Funktionsweise der Verbindung und des Protokolls auch mit zwei Delphi Programmen verfolgt werden. Das Server Programm simuliert hierzu das Verhalten des Mikrokontrollers. Die Oberfläche des Programms ist sehr einfach gehalten, um nicht durch eine aufwendige Gestaltung den Blick auf das Wesentliche zu erschweren.



Nach den Start des Programms müssen in zwei combo boxen der COM-Port und die Baudrate ausgewählt werden. Anschließend ist der Button [open port] freigegeben. Ein Klick auf [open port] öffnet dann die Verbindung. Zentrales Element ist ein Feld mit 26 Elementen für Floatzahlen (entsprechend A .. Z), in dem übertragene Werte (Parameter) gespeichert werden.

Im unteren Bereich werden die empfangenen Zeichen angezeigt.

Das Programm reagiert auf folgende Kommandos:

- Großbuchstaben Semikolon Float-Zahl CR LF (Beispiel X;1234,56 CR LF)
  Reaktion : Speicherung der Float-Zahl in einem Feldelement unter einem Index (A = 1, B = 2 …)
- Kleinbuchstaben CR LF (Beispiel x CR LF)
  Reaktion : Float-Zahl aus einem Feldelement senden. Index (a = 1, b = 2 …)
- Ausrufungszeichen CR LF (Beispiel ! CR LF)
  Reset ausführen, alle Feldelemente zu Null bzw. Voreinstellung setzen.
- Dollarzeichen CR LF (Beispiel $ CR LF)
  Alle Feldelemente dauerhaft speichern. Im Mikrokontroller wird dies später im EEPROM geschehen, das PC-Programm verwendet zur Simulation eine INI-Datei.
- Fragezeichen CR LF (Beispiel ? CR LF)
  Version der Firmware senden

5.2 Aufbau und Programmablauf

Das Serverprogramm ist in einer einzigen unit enthalten. Bei einer größeren Entwicklung würde man z.B. alle Funktionen die unmittelbar mit der seriellen Schnittstelle zusammenhängen in einem data modul zusammenfassen. Es kommt die Komponente **TComPort** zum Einsatz.

In den beiden OnChange Events der combo boxen werden die Portnummer und die Baudrate gesetzt. Die eigentlichen Setzfunktionen (*SetComPortN*r und *SetBaudRate*) kapseln den Zugriff auf die Routinen der TComPort Komponente. Damit braucht bei Versuchen mit alternativen Komponenten die eigentliche Programmlogik nicht geändert werden. Die

eintreffenden Zeichen der seriellen Schnittstelle lösen das OnRxChar Event der Komponente aus. Hier werden die eintreffenden Zeichen im String strRX gesammelt und auf da Zeichen LF untersucht. Wir das Zeichen LF gefunden, wird der String in *ParseRXStr* auf seine Bestandteile untersucht.

5.3 Parser

Nachdem die eintreffende Zeichenkette vollständig empfangen wurde, muss sie zerlegt und bearbeitet werden. Hierzu dient die Funktion *ParseRXStr*. Es müssen Klein- und Großbuchstaben, die zugelassenen Sonderzeichen sowie das Semikolon als Trennzeichen erkannt werden. Bei einem erkannten Semikolon soll dann der folgende Teil als Floatzahl interpretiert werden. Dabei ist es sinnvoll diesen Teilstring auf zulässige Zeihen zu überprüfen.

Auch hier werden nur Verfahren verwendet, die sich auch auf dem Mikrokontroller realisieren lassen, z.B. wird auf den Operator „in“ verzichtet und satt dessen werden die einzelne Zeichen in einer Schleife mit der Funktion *isValidValueChar* geprüft. Sofort offensichtlich ist die Zulässigkeit der Ziffern 0 ... 9 und eventuell der Vorzeichen + und -. Was ist aber mit einer Eingabe in wissenschaftlicher Notation, z.B. 1.234E-3. Also ist auch das Zeichen E und je nach Schreibweise auch e zulässig. Weiterhin muss ein Dezimaltrenner berücksichtigt werden, der wiederum, je nach sendendem System, entweder ein Punkt oder ein Komma sein kann. Dazu gleich noch eine Anmerkung: Der Mikrokontroller, bzw. BASCOM verträgt kein + am Anfang der Zahl und erwartet zwingend einen Punkt als Dezimaltrenner.

Die Funktion *ParseRXStr* führt nun die gewünschte Zerlegung durch und liefert das abgetrennte Kommando, den Index für die Speicherung und den Wert zurück. Erfolg bzw. ein Fehler wird durch ein entsprechendes Funktionsergebnis kenntlich gemacht.

Bei einer fehlerfreien Zerlegung wird dann in einer *case* Anweisung auf die empfangenen Kommandos reagiert und die entsprechenden Aktionen ausgelöst. Bei Kommandos die keine eigene Antwort erzeugen wird „ok“ gesendet, im Fehlerfall wird „error“ übertragen.

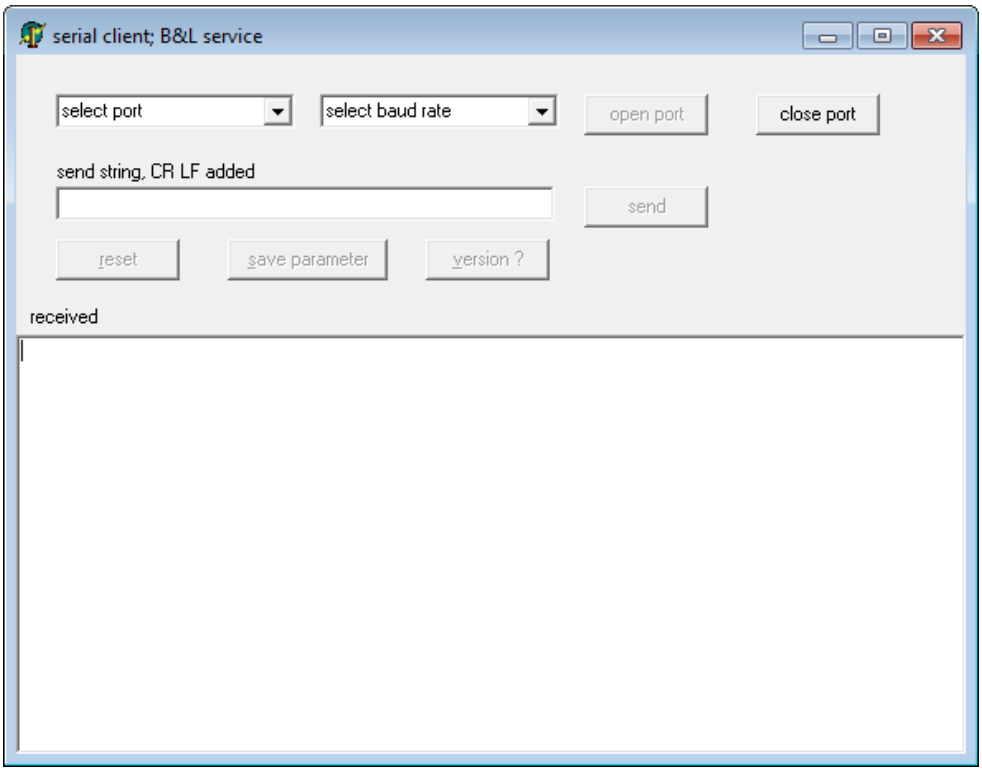
5.4 Simulierte Temperaturmessung

Die Temperaturmessung ist im Application.OnIdle Event angeordnet und wird über eine checkbox gestartet. Es wird ein AD-Wandler mit 8 Bit simuliert und unter Verwendung der Parameter offset und slope in eine reale Temperatur umgerechnet. Bei geöffnetem com port wird dieser Wert dann als String gesendet. Der zeitliche Abstand wird dabei über die Prozedur *delay* eingestellt. Auch während einer simulierten Messung reagiert das Programm auf Befehle.

6 Client Programm

Das Client Programm bietet im Wesentlichen die Funktionen eines Terminals, speziell zugeschnitten auf die vorliegende Anwendung.

6.1 Client Programm verwenden



Nach den Start des Programms müssen in zwei combo boxen der COM-Port und die Baudrate ausgewählt werden. Hier sei nochmals erwähnt, dass Client und Server auf gleiche baud rate aber unterschiedliche Port-Nummern eingestellt sein müssen. Anschließend ist der Button [open port] freigegeben. Ein Klick auf [open port] öffnet dann die Verbindung. Zusätzlich stehen drei Buttons [reset], [save parameter] und [version ?] zur Verfügung. Sie sind mit den bereits oben beschriebenen Spezialkommandos, also den Zeichen !, $ und ? belegt. Natürlich können diese Kommandos auch in das Eingabefeld getippt werden. Die erforderlichen Zeichen CR und LF werden bei einem Klick auf den Button [send] automatisch zugefügt.

6.2 Aufbau und Programmablauf

Der Aufbau des Client Programms ist wesentlich übersichtlicher als der Aufbau der Serversimulation, da hier kein Parser erforderlich ist. Einreffende Zeichen auf der seriellen Schnittstelle lösen das *OnRxChar* Event der Komponente aus. Hier werden die eintreffenden Zeichen im String strRX gesammelt und auf das Zeichen LF untersucht. Wird das Zeichen LF gefunden, kommt der String im Memofeld zur Anzeige.

7 Mikrokontroller Programm

Das Mikrokontroller Programm ist in BASIC geschrieben und wird mit dem BASCOM Compiler übersetzt. Auf die Art und Weise wie das Programm zum Kontroller übertragen

wird, soll hier nicht eingegangen werden, da dieser Vorgang stark von den vorhandenen Hilfsmitteln (Programmer) abhängt. Eine Demoversion der BASCOM IDE kann unter http://www.mcselec.com/index.php?option=com_docman&task=cat_view&gid=99&Itemid=54 heruntergeladen werden.

7.1 Initialisierung

Ganz zu Beginn des Programms müssen einige Einstellungen vorgenommen werden, die sich auf die zu Grunde liegende Hardware beziehen.

Die folgenden Zeilen stellen den Prozessortyp (hier ATMega32), die verwendete Quarzfrequenz und die Baudrate ein. Weiterhin wird die Interrupt-Routine für die serielle Schnittstelle zugewiesen und die Interrupts werden freigegeben. Die Einstellung des Prozessortyps und die Quarzfrequenz muss an die vorliegende Hardware angepasst werden.

```
$regfile = "m32def.dat"       ' used processor
$crystal = 14745600           ' used crystal frequency for processor
baud = 9600                   ' baudrate
on urxc Com1_Int              ' assign IRQ service
enable urxc
enable interrupts
```

Es folgt die Definition diverser Konstanten und Variablen. Vor den Start der Hauptschleife wird noch eine Versionskennung auf der seriellen Schnittstelle ausgegeben. Das hat den Vorteil, dass auch ohne funktionierende Kommunikation immer eine Ausgabe für einen ersten Test der seriellen Verbindung erzeugt wird.

7.2 Hauptschleife

Wie bei Mikrokontroller-Programmen üblich, erfolgt die Bearbeitung in einer großen Hautschleife, die mit dem *do* Statement beginnt und mit dem *loop* Statement endet. Hauptbestandteile der Schleife sind die Funktion *ParseRXStr* und das folgende *case* Konstrukt, welches das Ergebnis des Parsers auswertet.

7.3 Interrupt Behandlung

Jedes auf der seriellen Schnittstelle eintreffende Zeichen löst einen Interrupt aus, der in der Routine *Com1_Int* ausgewertet wird. Hier wird das eingetroffene Zeichen zunächst in der Variablen *b_udr1* zwischengespeichert und dann an den String *strRX* angehängt. Wird das Zeichen LF als Endzeichen erkannt, wird zur Benachrichtigung in der Hauptschleife die Variable *b_RxFlag* auf 1 gesetzt. Damit ist die Interruptbearbeitung beendet und sehr kurz gehalten. Die eigentliche Auswertung erfolgt dann am Kopf der Hauptschleife. Hier wird strRX gelöscht und *b_RxFlag* wieder auf 0 gesetzt.

7.4 Parser

Die Zerlegung und Auswertung des empfangenen Strings übernimmt die Funktion *ParseRXStr*. Die Aufgaben entsprechen der Beschreibung aus dem Abschnitt „Simulation auf dem PC“ Die beiden Funktionen *isValidValueChar* und *isValidIndexChar* untersuchen jeweils ein Zeichen auf seine Zulässigkeit. Dabei sorgt *isValidValueChar* auch gleich dafür,

dass Leerzeichen und Pluszeichen unterdrückt werden. Weiterhin wird ein Komma in einen Punkt gewandelt, da dies für die Umwandlung eines Strings in eine Zahl der korrekte Dezimaltrenner ist.

Die folgende Aufstellung zeigt die Reaktion des Parsers beim Empfang verschiedener Strings, wobei die ersten beiden Empfangsstrings ein Leerstring bzw. ein String mit einem Leerzeichen sind, die hier im Text natürlich nicht sichtbar sind.

```
invalid                     index 0 value 0.0                 command 0
invalid                     index 0 value 0.0                 command 0
invalid  X                  index 24 value 0.0                command 1
invalid  x;                 index 24 value 0.0                command 2
invalid  X;                 index 24 value 0.0                command 1
valid  x                    index 24 value 0.0                command 2
valid  X;1234.56            index 24 value 1234.559936522     command 1
valid  X;-1234.56           index 24 value -1234.559936522    command 1
valid  X;0.1234E-3          index 24 value 0.00012338         command 1
valid  X;+0.1234E-3         index 24 value 0.00012338         command 1
valid  X;+0.1234E+3         index 24 value 123.400001524      command 1
valid  X;-0.1234E+3         index 24 value -123.400001524     command 1
valid  X ;+ 0.12 34E + 3    index 24 value 123.400001524      command 1
```

7.5 Speicherung

Die beiden Funktionen *LoadParameterFromEEPROM* und *SaveParameterToEEPROM laden* bzw. speichern die Parameter vom / im EEPROM (engl. Abk. für electrically erasable programmable read-only memory, wörtlich: elektrisch löschbarer programmierbarer Nur-Lese-Speicher) des Prozessors. Damit stehen auch nach dem Aus- und Einschalten die geänderten Parameter zur Verfügung. Mit *ResetParameter* kann auf Befehl der Grundzustand wieder hergestellt werden. Im vorliegenden Beispiel werden dann die Parameter S (slope) und O (offset) sinnvoll vorbelegt.

7.6 Besonderheiten bei der Verarbeitung

In der Interruptbehandlung wird ein Flag gesetzt, um der Hauptschleife die Bereitstellung eines empfangenen Strings zu signalisieren. Beim nächsten Durchlauf erfolgt dann die Auswertung. Was geschieht aber, wenn in der Hauptschleife Teile mit langer Verarbeitungszeit enthalten sind? Dann verzögert sich die Reaktion des Kontrollers entsprechend. Da BASCOM kein Multitasking bereitstellt, ist diese Situation zunächst nicht zu vermeiden. Die Funktion *WasteTime()* steht hier stellvertretend für eine längere Verarbeitung und wartet 1 Sekunde. Während dieser Zeit wird permanent das Empfangsflag überwacht. Falls ein neuer String empfangen wurde, bricht die Wartezeit sofort ab und die Hauptschleife kann arbeiten. Außerdem zeigt das Funktionsergebnis den erfolgten Abbruch an. Nach dieser Methode können auch echte Verarbeitungen aufgebaut werden. Über ihr Funktionsergebnis signalisieren sie, ob ihre Verarbeitung unterbrochen wurde, oder ob die Ergebnisse verwendbar sind.

7.7 Temperaturmessung

Zur Demonstration wird hier eine (simulierte) Temperaturmessung herangezogen. Um das Beispiel unabhängig von einer realen Hardware zu machen, wird ein AD-Wandler mit 8 Bit Auflösung simuliert. Die Rohdaten haben damit einen Wertebereich von 0 bis 255 und sollen

hier eine Temperatur von –20°C bis 120°C repräsentieren. Eine Umrechnung auf reale Temperaturen soll bereits im Kontroller erfolgen. Hierfür werden, unter der Voraussetzung, dass die Wandlung eine lineare Kennlinie aufweist, zwei Werte benötigt. Es sind dies die Steigung und der Offset der Kennlinie. Unter idealen Voraussetzungen könnten beide Werte fest hinterlegt werden. In der Realität weisen aber Wandler und Geber immer Toleranzen auf. Zur Berücksichtigung dieser Toleranzen sollen die beiden Parameter ladbar gestaltet werden. Die Grundeinstellung ist dabei:

Steigung	= (120 – (-20)) / 255	= 0,549 °C/bit
Offset		= -20 °C
Temperatur	= AD-Bits * Steigung + Offset	

Die Funktion *GetTempRaw* erzeugt (simulierte) Werte zwischen 0 und 255, aufsteigend mit einem Inkrement von 4 Bit und dann Absteigend mit einem Dekrement von 2 Bit. Hierzu wird eine globale Variable (*b_TempSimulation*) verwendet. Die Funktion *CalculateTemp* berechnet unter Verwendung der hinterlegten Offset- und Steigungswerte die Temperatur in Grad Celsius. Die simulierte Temperaturmessung wird nur ausgeführt, wenn der Parameter T (also das Feldelement mit dem Index *parTempSimulation*) größer Null ist. Bei einem Reset wird das entsprechende Feldelement zu Null gesetzt. Vom Client muss also für den Start der Messung z.B. T;1 CR LF gesendet werden.

8 Praxistipp

Wenn bei einer seriellen Schnittstelle überhaupt nichts funktioniert geht man wie folgt vor:

- mit dem Gerätemanager nachschauen ob ein COM-Port angezeigt wird (bei USB seriell Wandlern dauert es eventuell lange bis die Installation eines Treibers abgeschlossen ist)
- einen (den) COM-Port merken
- Terminalprogramm mit diesem COM-Port starten, Flußsteuerung ausschalten
- RX und TX Leitung (an einem 9 poligen SUB-D ist das Pin 2 und Pin 3) verbinden
- Auf dem Terminal Zeichen eingeben. Diese Zeichen werden dann über TX gesendet und am Anschluss RX gleich wieder empfangen. Liegen keine Hardwaredefekte vor, müssen die gesendeten Zeichen (als Echo) im Terminal auftauchen.

Es kommt vor, dass sich ein USB-Treiber so „verklemmt“, dass erst ein aus- und einstecken des Wandlers zum Erfolg führt.

Da Hyperterminal in neueren Windows-Versionen nicht mehr enthalten ist kann als alternative Tera Term von hier beschafft werden. https://en.osdn.jp/projects/ttssh2/releases/

Achtung: In der Grundeinstellung sendet Tera Term nur CR nach einer Eingabe, soll CR LF gesendet werden, muss dies gesondert eingestellt werden.

9 Probleme bei der Verwendung von USB zu seriell Wandlern oder Blue-Tooth

Echte serielle Schnittstellen (COM-Port) sind mittlerweile in PCs und Notebooks eher selten zu finden. Ersatzweise kommen häufig USB zu seriell Wandler zum Einsatz. Über entsprechende Treiber emulieren sie einen COM-Port für das Betriebssystem. Auch im Gerätemanager wird ein COM-Port angezeigt. Leider gibt es eine Vielzahl von Konvertern mit recht unterschiedlicher Qualität vom Hardware und Treibern. Erschwerend kommt hinzu, dass ein erfolgreicher Einsatz in Zusammenhang mit einem bestimmten PC unter einem bestimmten Betriebssystem nicht unbedingt einen erfolgreichen Einsatz auf einem anderen PC garantiert. Diese Erkenntnisse sind keine Theorie, sondern resultieren aus langjährigen Erfahrungen von Anwendern und eigenen Einsätzen.

Grundsätzlich sollte man sich immer klarmachen, dass letztlich der Treiber für den Transport des seriellen Datenstroms über die USB-Schnittstelle verantwortlich ist. Dabei werden die seriellen Daten in einen USB-Datenstrom gewandelt, der dann übertragen wird. Bei nicht zu hohen Datenraten der emulierten seriellen Schnittstelle ist die USB-Übertragung immer „schnell genug“, auf ein exaktes Timing kann man sich aber nicht unbedingt verlassen.

Ähnliche Überlegungen gelten für das SPP Profil (serial port profile) bei Blue-Tooth.

https://de.wikipedia.org/wiki/Bluetooth-Profile