RS232 mit Eco-Physics-Protokoll (Anschlüsse und Protokollrahmen)

EPP_Protokollrahmen.doc, 2007-08-01 IS

Siehe auch unseren Text „Übertragungsprotokoll ‚Eco Physics' (Anwendungsbeispiele)".
Programmbeispiel: siehe das Excel-Add-In ‚cld2xls.xla' auf http://www.ecophysics.de
Der vorliegende Text beschreibt die elektrischen Eigenschaften der RS232-Schnittstelle und den Protokollrahmen des *Eco-Physics*-Kommunikationsprotokolls (CLD 7xy, CLD 8xy). Neben diesem proprietären Protokoll sind für die Geräte der Eco Physics AG noch Schnittstellen mit den folgenden Protokollen verfügbar:

- Bayern-Hessen-Protokoll für CLD 700 AL ba
- AK-Protokoll (Arbeitskreis der Automobilindustrie)

Inhaltsverzeichnis

Änderungen:

2002-12-27	Kapitel 4.4 (Die Programmiersprache ‚C' und das Nullbyte): neu
	Kapitel 1 (Anforderungen an das Kommunikationsprogramm): bearbeitet
2007-03-14	Kapitel 7 (Schnittstellenkabel): Beschreibung Zubehörkabel 9p-9p bearbeitet
	Kapitel 8 (Praktische Hinweise): bearbeitet
2007-07-30	Kapitel 4.2 (Blockprüfzeichen): bearbeitet
2007-08-01	Kapitel 9 (Leseroutine für Visual BASIC): neu

ECO PHYSICS GmbH
Zweigniederlassung Rösrath/Köln
Am Mittelscheid 4
D-51503 Rösrath
Fon +49 (0)2205 - 84049
Fax +49 (0)2205 - 910044
Homepage http://www.ecophysics.de
eMail ecophysics@ecophysics.de

Ingo-Kersten Schmuck
Technischer Service
eMail ingo.schmuck@ecophysics.de

1 Einleitung: Anforderungen an das Kommunikationsprogramm

Die Anforderungen des *Eco-Physics*-Protokolls und des *AK*-Protokolls sind zwar unterschiedlich kompliziert. Generell gilt aber: Sie lassen sich mit fertigen Kommunikationsprogrammen, die bloß das Senden und Empfangen von Textzeichen erlauben, nicht oder nur mit Einschränkungen erfüllen:

1. Der Zeichensatz des *Eco-Physics*-Protokolls in Befehl und Antwort enthält (auch) nichtalphanumerische Zeichen, nämlich ASCII-Steuerzeichen und ein Blockprüfzeichen, das beliebige Werte innerhalb der ASCII-Tabelle annehmen kann.
2. Das *Eco-Physics*-Protokoll erwartet am Ende des Befehlstelegramms ein Blockprüfzeichen, das zu bilden ist durch eine XOR-Verknüpfung der vorhergehenden Telegrammzeichen (Details: siehe unten). Bei feststehenden Befehlen immerhin kann man den Wert des Blockprüfzeichens „offline“ berechnen (zum Beispiel mit der Hilfe unserer Programme ComTest und cld2xls) und in eine Zeichenkette einbetten. Dabei hängt es von der Struktur des Befehls ab, ob das Bitmuster des Blockprüfzeichens in den Bereich druckbarer ASCII-Zeichen fällt.
3. Das Format des *AK*-Protokolls ist vergleichsweise weniger herausfordernd: Befehls- und Antworttelegramm sind einfach zwischen den ASCII-Steuerzeichen „STx“ und „ETx“ eingerahmt. Der Rest der Telegrammzeichen besteht aus druckbaren Zeichen, sofern nicht das „Don’t Care Byte“ davon abweichend eingestellt ist.
4. Die Datenabfrageroutinen (*AK* und *Eco Physics*) müssen zyklisch arbeiten: Jeder Datenabfragebefehl liefert nämlich nur einen einzigen Messwert beziehungsweise einen einzigen Messwertesatz bei den Mehrkanalgeräten.

Im allgemeinen lassen sich die Anforderungen der Protokolle nur mit frei zu programmierenden Systemen erfüllen.

2 Kommunikationsflusssteuerung

Der Steuerrechner kommuniziert mit dem Messgerät im „Master“-„Slave“-Verkehr: Der Steuerrechner (Master) sendet einen Befehl, dem eine Geräteadresse vorangestellt ist, das adressierte Gerät antwortet unmittelbar mit einem Telegramm. Das Messgerät kann nicht unaufgefordert aktiv werden.

3 Zeichenflusssteuerung

Die Schnittstelle des CLD 7xy/8xy unterstützt keine der Steuer- und Meldeleitungen. Diese werden bei der asynchronen Datenübertragung nicht unbedingt benötigt, weil dem Beginn eines Datenworts ohnehin ein vom Ruhezustand abweichender Sendeleitungszustand (das Startbit) vorangeht. Sie werden aber im allgemeinen benutzt, um den Sendefluss zu steuern, wenn der Empfänger zeitweilig nicht bereit ist. Verlangen die zur Wahl stehenden Schnittstellentreiber von der Gegenstelle die Bedienung der Meldeleitungen, so kann man dem eventuell durch das Erzeugen von Pseudosignalen entgegenkommen (siehe Kapitel 7, „Schnittstellenbeschaltung“).

Das Kommunikationsprogramm muss den Befehlstelegrammblock als ununterbrochene Zeichenfolge aussenden.
Es muss in der Lage sein, die mit konstanter Zeichengeschwindigkeit übertragenen Datenblöcke des Messgerätes in Empfang zu nehmen.

Flexibel nutzbare Kommunikationsroutinen organisieren den Datenverkehr mit der Hilfe von Datenpuffern. Dabei steuert der Schnittstellenbaustein durch Interrupts den rechtzeitigen Transfer der Zeichen vom Datenpufferbereich zum Schnittstellenregister und umgekehrt. Der eigentliche Datenverkehr und das Vorbereiten bzw. Weiterverarbeiten der Daten werden dadurch zeitlich entkoppelt.
Die Länge der Datentelegramme des CLD 7xy/8xy ist begrenzt. Ist der Empfangsdatenpuffer ausreichend dimensioniert, so ist das vollständige Einlesen der Telegramme im allgemeinen kein Problem.

Es ist allerdings bekannt, dass nicht alle „Industriestandard“-Rechner unter Windows 3.x und unter Windows 95 konstant fehlerfrei an der RS232-Schnittstelle arbeiten (beim Datenempfang gehen einzelne Zeichen verloren und es kommt eventuell zu Schnittstellen-Fehlermeldungen). Ein Kommunikationsprogramm sollte auf diesen Fall vorbereitet sein.

4 Der Zeichensatz des *Eco-Physics*-Protokolls

Der Zeichensatz des *Eco-Physics*-Protokolls in Befehl und Antwort enthält sowohl alphanumerische (druckbare) als auch nichtalphanumerische Zeichen, nämlich ASCII-Steuerzeichen und ein Blockprüfzeichen, das beliebige Werte innerhalb der ASCII-Tabelle annehmen kann.

4.1 ASCII-Steuerzeichen

Folgende Blockrahmen- und Quittungszeichen werden bei der Kommunikation mit dem CLD 7xx verwendet:

Ordnungszahl des Bitmusters dezimal	sedezimal („hexadezimal“)	Bedeutung (Kurzbezeichnung)
002_d	02_h	Start of text (STx)
003_d	03_h	End of text (ETx)
006_d	06_h	Acknowledge, Bestätigung (Ack)
021_d	15_h	Negative Acknowledge, Negative Rückmeldung (NAk)

4.2 Blockprüfzeichen (Block Check Character, BCC)

Das Blockprüfzeichen ist nach folgender Vorschrift zu bilden:
Die Telegrammzeichen müssen als ASCII-codierte Bitmuster vorliegen.
Führe mit den Bitmustern des ersten und des zweiten Zeichens eine XOr-Verknüpfung (XOr: „Antivalenz“, „Exclusiv-Oder“) in der Weise durch, dass jeweils die Bits gleicher Wertigkeit miteinander XOr-verknüpft werden. Verknüpfe das Ergebnis dieser Operation in der gleichen Weise mit dem nächsten Zeichen und fahre so fort bis zum letzten Zeichen (inklusive STx und ETx).

Ein Beispiel für Visual BASIC, in dem die Funktion Asc(Zeichen) das Bitmuster eines ASCII-Zeichens als Integerwert zurückgibt und Mid$(Zeichenkette, P, 1) das Zeichen von der Position P der Zeichenkette liefert.

```
Public Function fBCC(Zeichenkette As String) As Integer
   Dim BCC As Integer, P As Integer, L As Integer
   BCC = 0
   L = Len(Zeichenkette)
   If L > 0 Then
     For P = 1 To L
       BCC = BCC Xor Asc(Mid$(Zeichenkette, P, 1))
     Next P
   End If
   fBCC = BCC
End Function
```

Bemerkungen:
Das Ergebnis einer XOr-Quersumme ist unabhängig von der Reihung der Zeichen.

Die XOR-Verknüpfung von zwei gleichen Bitmustern liefert ein Ergebnis, in dem alle Bits Null sind.

Daraus folgt: Ergibt die Quersumme eines Antworttelegramms (das ein BCC enthält) nicht Null, so ist ein Übertragungsfehler bewiesen.
Hingegen ist eine korrekte Quersumme kein sicherer Beweis für eine fehlerfreie Übertragung. Ist nämlich eine gerade Anzahl an Zeichen falsch übertragen worden, so kann die XOr-Summe dennoch Null ergeben.

Das Bitmuster des BCC fällt nicht unbedingt in den Bereich alphanumerischer ASCII-Zeichen. 00_h kann vorkommen. Zum Beispiel kommt Stx 0 1 R R Etx auf die XOr-Summe 0. Generell ergibt sich in Befehlen mit der Geräteadresse ‚01’ immer dann die XOr-Quersumme 0, wenn die übrigen Zeichen des Befehls paarweise gleich sind.

Bei fixen Befehlen kann man sich den Wert des BCC einmal ausrechnen lassen (zum Beispiel mit der Funktion Direktbefehl unseres Excel-Add-Ins „cld2xls“ oder mit unserem Programm COMTest für MSDOS mit QBASIC) und dann in den Programmtext einfügen.

4.3 Die Antwort des CLD: Welche Zeichen kommen vor im Datenblock?

Ziffern, Großbuchstaben, Punkt als Dezimaltrennzeichen, Komma als Trennzeichen zwischen Datenfeldern.
Bit 7 ist nicht definiert. Es hat erfahrungsgemäß den Wert 0, wenn es übertragen wird (wenn die Wortlänge auf 8 Bit eingestellt ist).

Außerdem:
In den Bitmustern der Statusbytes (Antwort zum Befehl RS) ist Bit 6 fest gesetzt. Deren Ordnungswerte liegen damit im Bereich von 64_d (40_h) bis 127_d ($7F_h$).

Im CLD 70 Manual, Ausgabe 1995/1996 wurden beim RS-Befehl zusätzliche Statusbytes für Alarmgrenzwerte und digitale Schnittstelle dokumentiert („Option“), die den Wert 2_d (entsprechend ‘STX’) hätten annehmen können. Es wurde und wird jedoch keine Firmware ausgeliefert, die dieses spezielle Statustelegramm liefert.

4.4 Die Programmiersprache ‚C’ und das Nullbyte

Das Nullbyte (ASCII ‚Nul’) ist dasjenige Byte, in dem kein einziges Bit gesetzt ist. In mit der Programmiersprache ‚C’ geschriebenen Programmen und Betriebssystemen benutzt man es, um das Ende von Zeichenketten zu markieren. Das dem Nullbyte voran gehende Zeichen gilt damit als das letzte Zeichen der eigentlichen Zeichenkette. Mit ‚C’ geschriebene Routinen, die Zeichenketten kopieren oder zum Bearbeiten auslesen, brechen die Operation üblicherweise ab, wenn sie auf ein Nullbyte treffen. Wie wir gesehen haben (Kapitel 4.2), kann das BCC im Eco-Physics-Protokoll durchaus den Bytewert Null annehmen. Der Befehl ‚SendString’ der Programmpakete ‚RSAPI.DLL’ Version 1.1 und Version 1.22 für Windows (Autor: H.-J. Berndt) zum Beispiel sendet unter diesen Umständen nicht das BCC an den CLD, und der CLD antwortet dann nicht. Abhilfe: Den Befehl ‚SendByte’ verwenden und in C-Programmen das BCC nicht zusammen mit den anderen Zeichen in einer Zeichenkette speichern.

5 Das Telegrammformat des *Eco-Physics*-Protokolls

5.1 Telegrammstruktur

Die Telegrammstruktur und das Datenformat ist im Kapitel 8 der Bedienungsanleitung beschrieben. Die Codierung der verwendeten Blockrahmenzeichen und das Berechnen des obligatorischen Blockprüfzeichens (BCC, Block Check Character) haben wir ausführlicher in den Kapiteln 4.1 bzw. 4.2 des vorliegenden Textes beschrieben. Auf das Format der Befehlsdaten und auf das Fehlercode-Byte gehen wir nachfolgend noch ausführlich ein.

5.2 Geräteadresse

2 Dezimalziffern ASCII-codiert (‚00' ... ‚99').
Die Geräteadresse ist an der Handbedienungsoberfläche im Fenster <MENU> „Comm" einzutragen. Die Werte 0 bis 9 werden dort einstellig als ‚0' bis ‚9' angezeigt. Über die RS232-Schnittstelle sind sie jedoch als Zeichenfolge ‚00' bis ‚09' zu übertragen. Der Fabrikwert (Firmwarestandard) ist ‚01'.

5.3 Datenformat

Das Datenformat ist rigide.

5.3.1 Befehlstelegramm

Es sind gemäß ASCII codierte Zeichen zu übermitteln.

Numerische Daten sind im Dezimalformat als ASCII-codierte Ziffernfolgen zu senden.

In Fließkommazahlen ist ein Punkt als Dezimaltrennzeichen zu verwenden.

Der Einsatz eines Dezimaltrennzeichens (in Fließkommafeldern) ist optional. *Falls* jedoch ein Dezimaltrennzeichen übertragen wird, so muss mindestens eine Ziffer folgen (andernfalls quittiert der CLD 7xx mit der Kommunikationsfehlerart 4). Der Kalibrierreferenzwert „90,0 ppm" kann demnach in einem 5stelligen Datenfeld durch die Zeichenfolgen „00090" oder „090.0" ausgedrückt werden (Hinweis zur Datenfeldlänge: siehe unten).

Dem Dezimaltrennzeichen braucht keine Ziffer vorauszugehen.

Die Anzahl der Datenfelder und häufig auch die Datenfeldlänge (Zeichenanzahl) ist fix und befehlsspezifisch. Werden mehrere Datenfelder verlangt, so sind sie durch ein Komma zu trennen.
(Sofern die Datenfeldlänge fix ist, spezifiziert die Befehlsliste die je Datenfeld zu besetzenden Zeichenpositionen durch eine Reihe von Kleinbuchstaben, zum Beispiel *xxxxxx* . Der Dezimalpunkt zählt als ein Zeichen.)

Daten sind rechtsbündig in die Felder einzutragen. Nicht benötigte Zeichenpositionen am Anfang des Datenfelds sind durch Leerzeichen („Blank", ASCII 32_d) zu besetzen. In numerischen Feldern sind auch Nullen als führende Zeichen zugelassen.

Es gilt also der folgende Hinweis: Sofern die Dokumentation des Befehlssatzes eine bestimmte Datenfeldlänge (Zeichenanzahl)spezifiziert, muss diese eingehalten werden.

5.3.2 Antworttelegramm

Für das Aussehen des Antworttelegramms des CLD 7xx gelten im Großen und Ganzen die Regeln des Befehlstelegramms. In Datenfeldern sind nicht benötigte Zeichenpositionen führend oder folgend durch Leerzeichen besetzt. Falls mehrere Datenfelder vorhanden sind, werden sie durch Komma getrennt.

Bei den Messdatenrapporten ist folgendes zu beachten:

- Gewisse Gerätefehlermeldungen bringen das Gerät in den Zustand „Down". Falls das Gerät im Zustand „Down/Stand-by" ist (siehe Statusrapport RS), so kommt als Antwort auf den Messdatenrapportbefehl RD*n* nur ein Drei-Zeichen-Telegramm:
  <Ack><Fehlercode-Byte><Etx>. Und das Fehlercode-Byte meldet den Kommunikationsfehler Nr. 6 (= Nicht erlaubt im momentanen Instrumentenmodus).
- CLD 7xx: Die Datenfeldlänge ist fix.
- CLD 8xx: Die Datenfeldlänge der Messdatenrapporte ist nicht definiert.

Falls der Messwert verfügbar ist:
6 (ppm-Messgerät) oder 7 (ppb-Messgerät) Zeichen inklusive Dezimalpunkt und gegebenenfalls inklusive Minuszeichen. Von diesen 6 bzw. 7 Zeichen werden zur Zeit nur maximal 5 Zeichen für die Übermittlung des ppm- bzw. ppb-Zahlenwerts genutzt. Bei einer Zahl mit Minuszeichen wird die Anzahl der Ziffern um 1 reduziert. Mit führender Null, sofern Zahl < 1. Der Dezimalpunkt kann fehlen. Nicht benötigte Zeichenpositionen werden durch Anhängen von Leerzeichen (ASCI 32_d) aufgefüllt. Man sollte aber nicht ausschließen, dass auch führende Leerzeichen möglich sind. Beispiele: -0.12, 0.123, 1.234, 12.34
Achtung:
Der Befehl RD*n* liefert nicht ppm- bzw. ppb-Messwerte, sondern Rohsignale, solange an der Handbedienungsoberfläche des Instruments <TEST> „Counts“ aktiviert ist. Rohsignale ergeben maximal 5 Ziffern (ppm-Messgerät) bzw. maximal 6 Ziffern (ppb-Messgerät).

Falls der Messwert wegen abweichender Messart nicht verfügbar ist:
nur ein Sternchen „*“ (dies gilt für den CLD 8xy; beim CLD 7xy gilt anderes)

5.4 Kommunikationsfehlermeldungen, Fehlercode-Definitionen

- Der CLD antwortet nicht, wenn die Geräteadresse abweicht oder solange nicht, wie auf das Blockrahmenzeichen „ETx“ kein weiteres Zeichen folgt (regulär: das BCC).
- Der CLD antwortet mit einer „NAk“-Meldung (Kapitel 4.1), wenn eine der nachfolgenden Bedingungen eintritt:
  Blockprüffehler: Das gelesene Blockprüfzeichen (Kapitel 4.2) passt nicht zu den empfangenen Daten.
  Befehlsüberlauf: Der CLD hat das Schlusszeichen „ETx“ und das „BCC“ des vergangenen Befehlstelegramms verpasst und empfängt nun einen neuen gültigen Befehl.
- In allen anderen Fällen antwortet der CLD mit einer „Ack“-Meldung (Kapitel 4.1). „Ack“-Meldung bedeutet nicht unbedingt, dass der empfangene Befehl ausgeführt worden ist (siehe Kommunikationsfehlercode).
- Jede Antwort enthält ein Fehlercode-Byte.
  Definitionen des Fehlercode-Bytes:

Bit 0...3: Kommunikations-Fehlercode
0 kein Fehler (Befehl ausgeführt)
1 Blockprüffehler *)
2 Befehlsüberlauf (Blockrahmenfehler) **)
3 Ungültiger Befehl (nicht definierter Befehlscode)
4 Ungültige Operation (ungültige Befehlsdaten)
5 ---
6 Befehl nicht erlaubt im momentanen Instrumentenmodus
*) NAK-Telegramm. Der jüngste Befehl wurde nicht ausgeführt.
**) NAK-Telegramm. Die jüngsten zwei Befehle wurden nicht ausgeführt.

Zum Aufklären oder Vorbeugen der Fehlermeldung „Befehl nicht erlaubt im momentanen Instrumentenmodus“ muss anhand des Statusrapports „RS“ geprüft werden, ob eine der Befehlsvoraussetzungen verletzt ist (siehe Anmerkungen zu den Befehlen im Kapitel 8 der Bedienungsanleitung). Gerätefehler oder vorübergehende Netzunterbrechungen können den Gerätestatus nachhaltig ändern. Und ist das Gerät in Lokalbedienbereitschaft, so kann der Gerätestatus auch durch Operationen an der Handbedienungsoberfläche geändert werden.

Bit 4: Geräte-Warnungsbit
Dieses Bit ist gesetzt, wenn eine Gerätewarnung (W-*xy*) ansteht. Auf den Befehl „RS“ („Report Status“) wird unter anderem der Warncode mit der höchsten Priorität (CLD 7xy) bzw. eine Liste mit allen anstehenden Warnungen (CLD 8xy) zurück gesendet.

Bit 5: Geräte-Fehlerbit
Dieses Bit ist gesetzt, wenn ein Gerätefehler (E-*xy*) ansteht. Auf den Befehl „RS“ („Report Status“) wird unter anderem der Fehlercode mit der höchsten Priorität (CLD 7xy) bzw. eine Liste mit allen anstehenden Fehlermeldungen (CLD 8xy) zurück gesendet.

Bit 6: Immer 1

Bit 7: nicht definiert

6 Kommunikationsroutinen (Forderungen an ...)

Flexibel nutzbare Kommunikationsroutinen organisieren den Datenverkehr mit der Hilfe von Datenpuffern. Dabei steuert der Schnittstellenbaustein selbst durch Interrupts den rechtzeitigen Transfer der Zeichen vom Datenpufferbereich zum Schnittstellenregister und umgekehrt. Der eigentliche Datenverkehr und das Vorbereiten bzw. Weiterverarbeiten der Daten sind dadurch zeitlich entkoppelt.
Beim Betriebssystem „MS-DOS“ müssen Interrupt-Kommunikationsroutinen und Datenpuffer vom Anwendungsprogramm selbst mitgebracht beziehungsweise eingerichtet werden. Fertig programmierte Module gehören zum Lieferumfang von Programmiersprachenpaketen oder sind separat erhältlich. Andere Betriebssysteme bringen entsprechende Funktionen bereits mit.

Das Datenempfangsprogramm muss für wechselnde Bedingungen gewappnet sein:
- Ausbleiben einer Antwort,
- Unvollständigkeit eines Telegramms,
- Fehlermeldungen des Computers (Schnittstellenfehler),
- Fehlermeldungen des CLD 7xx

Das Kommunikationsprogramm sollte System unabhängig sein, das heißt, es sollte nicht durch eine Änderung der Arbeitsgeschwindigkeit des Rechners oder der Übertragungsgeschwindigkeit gestört werden.

6.1 Identifizieren des Telegrammendes

Die Länge der Datentelegramme des CLD 7xy/8xy ist begrenzt.
Damit nicht jede auf den Empfangsdatenpuffer ausgeübte Leseoperation erst durch ein „Time-out“ beendet wird, wird ein Kriterium für das letzte Zeichen eines Telegramms benötigt.
Die Entleerung des Empfangsdatenpuffers ist kein hinreichendes Kriterium für das Ende eines Telegramms, weil das Ausleseprogramm ja schneller als der Sender (der CLD 7xx) sein kann.

Kriterium Telegrammendezeichen (ETx)
Holt man Zeichen für Zeichen einzeln aus dem Empfangspuffer, so kann man dabei einfach auf das Eintreffen des Telegrammendezeichens (ETx) prüfen.

Kriterium Zeichenanzahl:
Die Anzahl der Zeichen im Antworttelegramm folgt zwar festen Regeln im *Eco-Physics-Protokoll*, sie ist jedoch nicht nur befehlsabhängig, sondern sie hängt auch vom Fehlerstatus des CLD 7xx ab.

Beim AK-Protokoll ist die Anzahl der Zeichen begrenzt, aber im allgemeinen kaum vorauszusehen.

6.2 Ausleseroutinen

Die Übertragung eines Zeichens vom CLD 7xx in das Empfangsregister des Schnittstellenbausteins eines Steuerrechners beansprucht mehr Zeit als der Transfer eines Zeichens vom Schnittstellenbaustein in den Datenpuffer des Betriebssystems. Starten wir die Auswertung des Telegramms erst nach dem Ende der Übertragung, so bleibt verfügbare Rechenzeit ungenutzt, denn die Zeit bis zur Bereitstellung des nächstens Zeichens kann zur Auswertung der bereits eingegangenen Zeichen genutzt werden.. Der parallele Ablauf von Datenübertragung und Auswertung ist günstig, wenn der Zeitverzug bis zum Vorliegen des Auswertungsergebnisses minimiert werden soll. Auf der Kehrseite steht der relativ hohe Aufwand zur Synchronisierung der Prozesse. Der Rechner muss also hinreichend schnell sein, damit die Parallelisierung die erwartete Beschleunigung bringt (Kapitel 6.2.1).
Sind wir bei gegebener Messdatenrate vorrangig an der Maximierung der für die Auswertung verfügbaren Rechnerzeit interessiert, so ist es günstiger, während der Übertragung eines Telegramms vom CLD 7xx zum Empfangsdatenpuffer des Betriebssystems die Daten des vorhergehenden Zyklus auszuwerten (Abschnitt 6.2.2).

6.2.1 Das sukzessive Auslesen des Empfangspuffers während der Datenübertragung

Das Auslesen des Puffers während der Datenübertragung erfordert einigen Synchronisierungsaufwand:
- Weil das Leseprogramm nicht wissen kann, ob die Datenübertragung mit der Ausleseprozedur Schritt hält, muss für jedes Zeichen eine Warteschleife initialisiert werden.
- Weil das Ausleseprogramm nicht wissen kann, ob die Übertragung tatsächlich beginnt und ein reguläres Ende findet, muss vor dem Eintreten in die erste Warteschleife ein Timer für den Notausgang initialisiert werden.

Man kann einen Timer einmal vor dem ersten Leseversuch mit einer für ein ganzes Antworttelegramm ausreichenden Frist initialisieren oder wiederholt mit kürzerer Frist für jedes einzelne Zeichen. Die zweite

Methode hat den Vorteil, dass nach dem Erkennen von Fehlern in frühen Phasen der Datenübertragung die Wiederholung früher eingeleitet werden kann, sofern sich die Wartefrist hinreichend fein an die Übertragungsgeschwindigkeit anpassen lässt.
Ein Merkmal des sukzessiven Lesens und Prüfens der Zeichen ist übrigens, dass keine Annahme über die Anzahl der zu erwartenden Zeichen gemacht werden muss, wenn ein Textendezeichen definiert ist wie beim *Eco-Physics*- und beim *AK*-Protokoll. Speziell beim *Eco-Physics*-Protokoll aber, bei dem ja die Länge der Antworttelegramme festen Regeln folgt, kann es von Vorteil sein, wenn das Leseprogramm vorab weiß, wie viele Zeichen es erwarten soll (siehe Kapitel 6.2.2).

6.2.2 Das Auslesen des Empfangsdatenpuffers nach dem Ende der Übertragung

Ist dem Programm die Anzahl der zu erwartenden Zeichen bekannt (bei Telegrammen des *Eco-Physics*-Protokolls sind sie bedingt bekannt, siehe unten), so kann das Ende der Datenübertragung anhand der Anzahl der im Puffer eingetroffenen Zeichen verfolgt werden. Dafür muss die Warteschleife lediglich einmal initialisiert werden. Im folgenden Beispiel wird das eingegangene Telegramm 'en bloc' in einer Textvariablen zur späteren Auswertung gespeichert:

```
Erste Befehlssendung initialisieren
Antwortfrist (je nach „Baudrate“ und Antwortlänge) initialisieren
Erste Antwort abholen und zwischenspeichern oder „Time out“
Zeitpunkt „nächster Befehl“ berechnen und merken

Wiederhole....
    Falls Zeitpunkt „nächster Befehl“ erreicht, dann
        Befehlssendung initialisieren (Übertragung im Hintergrund)
        Zeitpunkt „nächster Befehl“ berechnen und merken
        Antwortfrist (je nach „Baudrate“ und Antwortlänge) initialisieren
        Antwort des vorhergehenden Zyklus auswerten
        Irgend etwas nützliches tun
        Antwort abholen (durch Hintergrundprogramm eingelesen und in einem
            Datenpuffer bereitgestellt) und zwischenspeichern oder „Time out“
    Irgend etwas anderes nützliches tun
...bis Ende der Messung

Antwort des letzten Zyklus auswerten
```

Man sieht, Telekommunikation und Auswertung des vorhergehenden Zyklus können parallel verlaufen, sofern die Kommunikation mit dem Schnittstellenbaustein im Hintergrund verläuft.

Als Antwort auf einen Datenabfragebefehl kommt entweder ein Datentelegramm bekannter Länge oder ein Fehlertelegramm bekannter Länge (3 Zeichen). Es kann aber anders kommen, als zu erwarten, deshalb muss eine Schleife eingerichtet werden, die mit dem regulären Ende des Telegramms oder dem Ablauf der Antwortfrist terminiert. In unserem Fall sollte also der in dem Beispiel oben aufgelistete Programmblock „Antwort abwarten und zwischenspeichern“ wie folgt aussehen:

```
Die ersten 3 Zeichen erwarten und identifizieren oder „Time out“
Falls o.k., dann
    Fehlercode-Zeichen separieren und auswerten
    Falls drittes Zeichen = STX (also Datenblock folgend), dann
        Restliche n Zeichen erwarten, separieren und speichern oder „Time out“
```

Die Implementierung der Zeile „Restliche *n* Zeichen erwarten.....“ muss folgendes berücksichtigen:
Soll sie als eine für Antworten unterschiedlicher Struktur verwendbare Routine realisiert werden, so muss Zeichen für Zeichen geprüft werden, um die Struktur des Telegramms aufzuklären. Der Ablauf gleicht dann sehr dem in Abschnitt 6.2.1 (sukzessives Auslesen) beschriebenen Muster.

Minimal wird der Synchronisierungsaufwand, wenn die Auswerteroutine für eine feststehende Antwortstruktur spezialisiert ist, weil dann lediglich einmal geprüft werden muss, ob die zu erwartende Anzahl an Zeichen eingegangen ist.

7 RS232 Schnittstellenkabel

Schnittstelle des CLD 7xy /8xy:

- D-Sub 9polig, männlich
- Daten-End-Einrichtung (DEE)
- Kontaktbelegung: siehe CLD 7xy/8xy Manual
  Die für RS485-Signale reservierten Kontakte sind nicht belegt auf Schnittstellenkarten mit Standardbestückung.
- DTR ist nach dem Einschalten des Geräts permanent „ein".
- RTS wird nach dem Einschalten des Geräts eine Sekunde lang „aus" gesetzt, danach permanent „ein".
- DSR und CTS werden nicht ausgewertet.

Der im Fenster <MENU> „Comm." wählbare Modus „RS485" wird nicht unterstützt.

Die Belegung des 9poligen Steckers (siehe Anschlussschema im Kapitel 8 der Gebrauchsanleitung des CLD 7xy/8xy) ist partiell kompatibel mit dem „IBM-PC-AT-RS232-Standard".
Die Funktion der *RS232*-Schnittstellenkontakte ist gemäß der Konvention für Datenendeinrichtungen (DEE, englisch DTE für „Data Terminal Equipment") definiert, das heißt:

RxD ist Dateneingang,
TxD ist Datenausgang,
RTS ist Steuerausgang,
CTS ist Meldeeingang,
DTR ist Steuerausgang

Obwohl also RTS, CTS und DTR am Schnittstellenstecker des CLD 7xy/8xy elektrisch definiert sind, unterstützt oder benutzt das Schnittstellenprotokoll keine dieser Steuer- und Meldekontakte, allerdings setzt der CLD 7xy/8xy den RTS-Ausgang während der ersten Sekunde nach dem Einschalten der Energieversorgung *aus*, danach dauernd *ein,* und DTR ist feststehend eingeschaltet. Somit macht es Sinn, den DTR- oder den RTS-Ausgang des CLD für eine Einschaltkontrolle an einem der Meldeeingänge (CTS, DSR, RI) des Steuerrechners auszuwerten.
Zwei weitere Kontakte sind für Signale gemäß RS485-Standard reserviert.

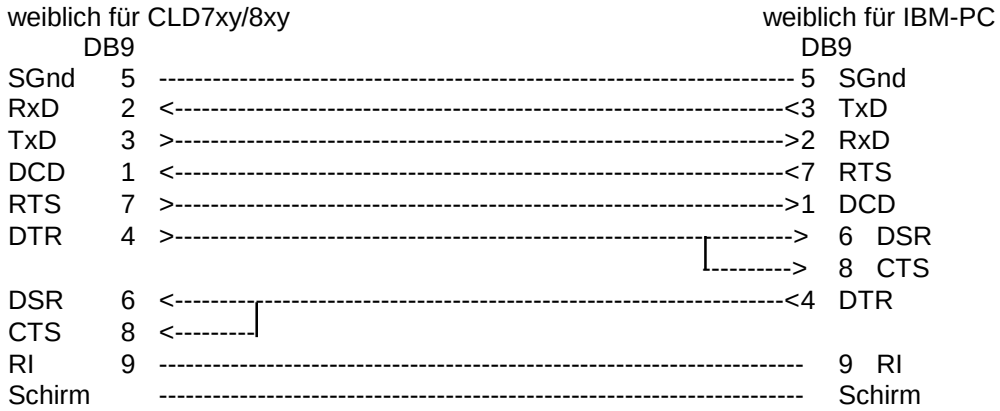
Die folgenden Abbildungen zeigen die Schaltungen der früher (9p-25p) und aktuell (9p-9p) im Zubehör gelieferten Schnittstellenkabel. Sie stellen die Schnittstellensignale so an einer Federbuchse (= weiblich) bereit, wie es für die Zusammenarbeit mit einem PC erforderlich ist: Der Datenausgang (TxD) des CLD 7xy/8xy ist mit dem RxD-Pol der Federbuchse verbunden, und der Meldeausgang RTS wird auf den CTS-Pol geführt.

9p/25p-Seriell-Nullmodemkabel (*Eco-Physics*-Schnittstellenkabel alt):

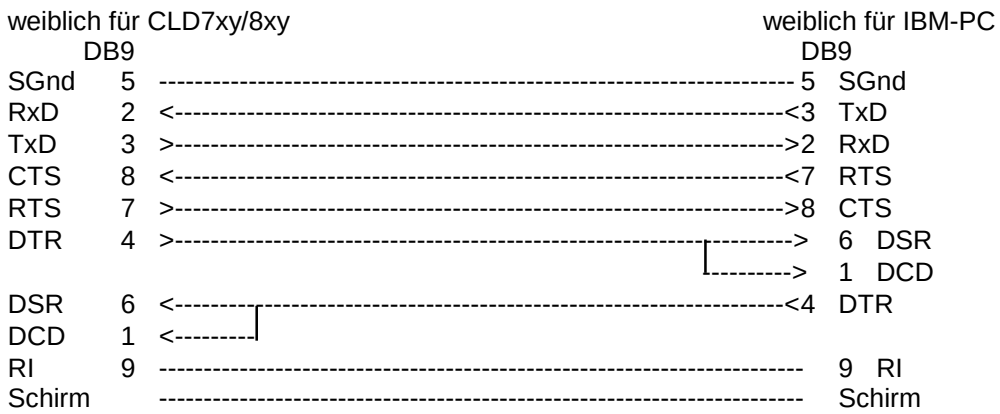
weiblich für CLD7xy/8xy			weiblich für IBM-PC	
DB9			DB25	
SGnd	5	----------	7	SGnd
RxD	2	<----------	<2	TxD
TxD	3	>----------	>3	RxD
RTS	7	>----------	>5	CTS
CTS	8	<----------	<4	RTS
Schirm		----------		Schirm

Für die Verbindung zu Steuerrechnern mit verkleinerten RS-232-Steckern (die 9-polige Schnittstellenbuchse gemäß IBM-PC-AT-RS232-Standard, z.B.) wird zusätzlich ein handelsüblicher Adapter („AT-RS232-Adapter") benötigt. Dieser Adapter darf nicht auch mit einer „Nullmodem"-Funktion ausgestattet sein, falls er mit dem im Zubehörsatz des CLD gelieferten Schnittstellenkabel kombiniert wird. Handelsübliche „RS232-Druckerkabel" z.B. dürfen demnach nicht mit dem *Eco Physics*-Schnittstellenkabel kombiniert werden.

9p/9p-Seriell-Nullmodemkabel (*Eco-Physics*-Schnittstellenkabel neu, Fabrikat „roline“):



9p/9p-Seriell-Nullmodemkabel (*handelsübliche Variante*):

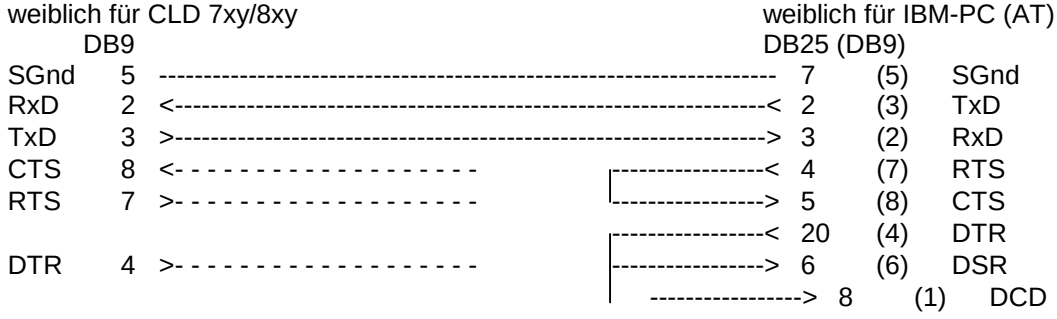


Grundsätzlich reicht schon eine Drei-Draht-Verbindung (die beiden Datenleitungen mit der gemeinsamen Rückleitung via Betriebserde, SGnd).

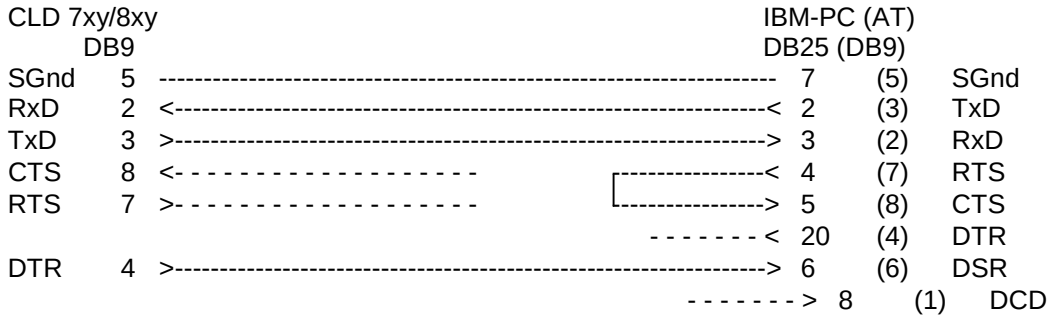
Die Drei-Draht-Verbindung macht eine Kommunikation allerdings nur dann möglich, wenn das auf dem Steuerrechner laufende Programm nicht die Aktivierung von Meldeleitungen durch den CLD erwartet (das heißt: voraussetzt). Die meisten Telekommunikationsprogramm-Module lassen sich auf diese Konstellation einstellen. Dasselbe gilt für den Befehlssatz der BASIC-Interpreter der Firma Microsoft.

Benutzen Sie ein fertiges Kommunikations-Software-Modul, dessen Eigenschaften nicht bekannt sind oder nicht geändert werden sollen, so kann es hingegen zu Problemen kommen.
Erwartet das Kommunikationsmodul lediglich die Aktivierung des CTS-Pols als Voraussetzung für den Sendebeginn, so ist das Kabel des Zubehörsatzes gerade ausreichend.
Verlangt das Programm weitere Meldeleitungen oder ist nicht bekannt, nach welchen Regeln es arbeitet, so kommen Sie vielleicht mit einer der folgenden Schaltungen aus:

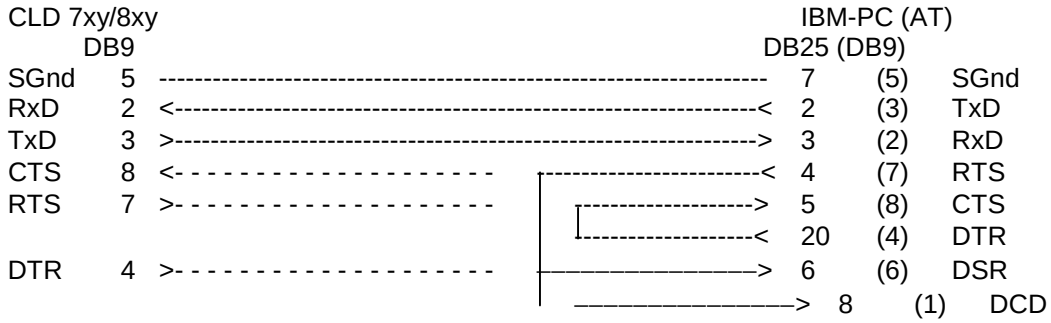
Seriell-Nullmodemkabel mit Pseudo-Meldeleitungen
Variante 1a:



Variante 1b:
Diese Variante ist kompatibel zu der in DIN 66 258 Teil 1 beschriebenen Festlegung der VS-Schnittstelle (Voltage Serial). Literatur: Harald Schumny, Rainer Ohl; Handbuch digitaler Schnittstellen, Vieweg Verlag, erste Auflage 1994.



Variante 2:
Auch die folgende Verdrahtung wird in der Literatur vereinzelt vorgeschlagen:



Der Computer kann also selbst die verlangten Meldesignale erzeugen. Wird aber die Empfangsbereitschaft erst durch 'DCD aktiv' hergestellt und macht das Kommunikationsprogramm das Erscheinen von DCD zur Voraussetzung für das Setzen von DTR, so muss das DCD-Signal vom CLD-RTS-Ausgang oder von einer anderen Gleichspannungsquelle abgeholt werden.
Auf keinen Fall dürfen Signalausgangs-Kontakte miteinander verbunden werden.

Datenbrücke für den Test der Schnittstelle des PC
Mit der Funktion ‚Com-Test' des Fensters ‚Direktbefehl' kann die Funktion der RS232-Schnittstelle des PC geprüft werden, nachdem Sie an Stelle des Messgerätes ein Kabel mit Datenbrücke angeschlossen haben (9-polige und 25-polige Stecker: Verbinde die Kontakte Nr. 2 und 3, das heißt: Verbinde die Leitung RD mit der Leitung TD). Damit empfängt der Computer Daten, die er selbst gesendet hat.

Anschließen von 25-poligen RS232-Prüfzwischensteckern

... an 25p-9p-Nullmodemkabel:
PC-RS232 (9p männlich) >
AT-RS232-Adapterkabel (9p weiblich – 25p männlich) >
RS232-Prüfzwischenstecker (25p weiblich – 25p männlich) >
Nullmodemkabel (25p weiblich – 9p weiblich) >
CLD-RS232 (9p männlich)

Die Anschlussfolge kann umgedreht werden: PC und CLD können den Platz tauschen.

... an 9p-9p-Nullmodemkabel:
PC-RS232 (9p männlich) >
AT-RS232-Adapterkabel (9p weiblich – 25p männlich) >
RS232-Prüfzwischenstecker (25p weiblich – 25p männlich) >
Zwischenstecker 25p weiblich – 25p weiblich („Gender-Changer") >
AT-RS232-Adapterkabel (25p männlich – 9p weiblich) >
Zwischenstecker 9p männlich – 9p männlich („Gender-Changer") >
Nullmodemkabel (25p weiblich – 9p weiblich) >
CLD-RS232 (9p männlich)

Die Anschlussfolge kann umgedreht werden: PC und CLD können den Platz tauschen.

8 Praktische Hinweise zur RS232-Kommunikation

Ein Nullmodemkabel mit Minimalausstattung (RD und TD überkreuzt, SGnd verbunden) genügt für die Kommunikation mit dem CLD.

Die **Schnittstellenparameter** und **Protokollparameter** der Kommunikationspartner müssen übereinstimmen, andernfalls kommt die Kommunikation nur mit Fehlern oder gar nicht zustande. (Ausnahme: Stopbits können auf beiden Seiten unterschiedlich eingestellt sein.) Die Parameter des CLD 7xx sind unter der Taste „MENU“ im Fenster „Comm.“ einzustellen. Zu den veränderlichen Parametern gehört hier auch die Geräteadresse des CLD, welche ja im Befehlstelegramm zu übertragen ist. Wurde ein Master-Reset (Fabrikeinstellungen laden) am CLD 7xy/8xy durchgeführt oder kam es aus anderen Gründen zur Fehlermeldung E01, so müssen die Sollparameter wiederhergestellt werden, sofern sie von den Fabrikstandardwerten abweichen.
Fabrikstandardwerte:
9600 Baud, Wortlänge = 7 bit, Stopbit = 1, Paritätsbit = off (d.h. keines), Geräteadresse = 1.

Mitunter wird die RS232-Schnittstelle blockiert, wenn das Messgerät erst nach dem Start einer Messaufzeichnung an den Computer angeschlossen wird. Beenden Sie cld2xls gegebenenfalls und starten Sie es erneut.

Manche USB-RS232-Adapter geben falsche Rückmeldung über die Verfügbarkeit der RS232-Schnittstelle, wenn sie am Rechner ein- oder ausgesteckt worden sind, während der Rechner ausgeschaltet war. Durch Ein- und Ausstecken des USB-Kabels und Drücken der Schaltfläche „Com-Port-Test“ des Editors der COM-Port-Liste der Messeinstellungen können Sie verfolgen, unter welcher COM-Port-Nummer sich der Adapter meldet.
Zudem wurde beobachtet, dass RS232-Adapter nach dem Aus- und Einstecken des USB-Kabels eventuell zu einem anderen COM-Port wechselten. Die zuvor genutzten Einstellungen in cld2xls waren damit nutzlos.

Der CLD antwortet immer, sofern die Geräteadresse übereinstimmt und der Protokollrahmen komplett erkannt worden ist. Der CLD antwortet nicht, solange er das BCC nicht erhalten hat. Antwortet der CLD nur bei bestimmten Befehlen nicht, und gibt er dann beim nächsten (anders lautenden) Befehl eine Kommunikationsfehlermeldung, so prüfen Sie doch, ob Ihr Kommunikationsprogramm das Telegramm in jedem Fall komplett überträgt (Kapitel 4.4).

Prüfen der elektrischen Funktion mit Hilfe eines RS232-Prüfzwischensteckers
Der Prüfadapter wird in die Signalleitung eingefügt und informiert mit der Hilfe von Leuchtdioden (*Light Emitting Diode*, LED) über die elektrischen Pegel:

	RS232-Tester LED-Aktivität	**Datenleitung**	**Steuer-/ Meldeleitung**
Pegel „H“	Farbe 1: Rot (oder Grün) *	0	Ein
Pegel „L“	Farbe 2: Grün (oder Rot) *	1	Aus

*) Farbe je nach Fabrikat des Steckers

Stopp-Pegel auf der Datenleitung: „L“
Startbit auf der Datenleitung: „H“

Pegeldefinitionen
Empfangspegel „H“: > 3 Volt (Sendepegel: max. 15 V, real ca. + 8 Volt an Schnittstellen mit MAX232)
Empfangspegel „L“: < -3 Volt (Sendepegel: min. -15 V, real ca. - 7,4 Volt an Schnittstellen mit MAX232)
Bei ungültigen Pegeln (zwischen -3 Volt und + 3Volt) leuchten die LED nur schwach oder gar nicht.

Der Zustand der Schnittstelle des CLD
Hängt der RS232-Prüfstecker an einem 1:1 Verbindungskabel, das *alle* vom CLD 7xx / 8xx (mit Fluoreszenzdisplay) bereitgestellten Signale übergibt, *ohne* sie zu vertauschen (das ist also ein Kabel ohne Nullmodem-Funktion), so ist
TD (TxD) = L
und bei den Geräten mit Fluoreszenzdisplay zusätzlich:
RTS = H (beim Einschalten des CLD erst L, H nach ca. einer Sekunde),
DTR = H
Geräte mit AK-Bedienungsoberfläche liefern eventuell kein RTS und DTR.

Ist der RS232-Prüfstecker *über das alte **9p-25p**-Eco-Physics-Schnittstellenkabel* allein mit dem CLD 7xx / 8xx (nicht aber mit dem Computer) verbunden, so sollte sein

RD (RxD) = L,

und bei den Geräten mit Fluoreszenzdisplay zusätzlich:

CTS = H (beim Einschalten des CLD erst L, nach ca. einer Sekunde H).

Geräte mit AK-Bedienungsoberfläche liefern eventuell kein CTS.

Ist der RS232-Prüfstecker *über das **9p-9p**-Eco-Physics-Schnittstellenkabel* allein mit dem CLD 7xx / 8xx (nicht aber mit dem Computer) verbunden, so sollte sein

RD (RxD) = L,

und bei den Geräten mit Fluoreszenzdisplay zusätzlich:

DCD = H (beim Einschalten des CLD erst L, nach ca. einer Sekunde H)

DSR = H (schwach leuchtend)

CTS = H (schwach leuchtend)

Geräte mit AK-Bedienungsoberfläche liefern eventuell neben RD keine weiteren Signale.

Für *cld2xls* ist weder DCD, noch DSR, noch CTS erforderlich.

Der Zustand der Schnittstelle des PC

Trennen Sie jetzt die Verbindung zum CLD und schließen Sie den RS232-Checker über ein Adapterkabel ohne Nullmodem-Funktion an den PC an. Es sollte dann je eine LED der Leitungen TD (TxD), RTS und DTR aufleuchten. Grundzustand: L, solange kein Kommunikationsprogramm die Kontrolle über die RS232-Schnittstelle übernommen hat.

Achtung: Pulsiert die Leitung TD zeitweise zwischen L und H oder hat RTS oder DTR den Zustand H, bevor in cld2xls eine Messaufzeichnung gestartet worden ist, so bedeutet das, dass die RS232-Schnittstelle schon durch ein anderes Programm belegt ist und cld2xls nicht ohne weiteres den Zugriff erlangen wird. Verantwortlich dafür kann ein im Hintergrund arbeitendes Programm sein, das ständig eine der oder alle RS232-Schnittstellen nach angeschlossenen Geräten absucht. Beispiele für solch ein Programm:

- Treiber für Mäuse oder Grafiktabletts mit RS232-Anschluss
- Programme für den Datenaustausch mit Mobiltelefonen
- Programme für den Datenaustausch mit Computern

Manche (ältere) Multifunktionsprogramme haben Kommunikationsmodule, die ständig eine der RS232-Schnittstellen in Beschlag nehmen.

Die Zusammenarbeit von PC und CLD

Sind PC und Messgerät über das *Eco-Physics*-Schnittstellenkabel und den RS232-Checker *miteinander* verbunden, so sollten schließlich die LED der genannten 5 bis 7 Leitungen (je nach Nullmodemkabel) gemeinsam leuchten.

Beim Start der Messaufzeichnung wechselt RTS von L nach H. Während der Messdatenübertragung ist zu sehen, wie die Datenleitungen RD und TD zwischen L und H pulsieren. Bei niedriger Datenrate (300 Baud) ist dieser Effekt besser zu erkennen als bei hoher (9600 Baud).

9 Anhang: Leseroutine für Visual BASIC

Die nachfolgende Prozedur liest ein Antworttelegramm des CLD *Byte für Byte* aus dem RS232-Empfangsdatenpuffer, prüft die Struktur der Antwort und stellt gefundene Daten für andere Prozeduren bereit.

Sie nutzt zwei Funktionen der "RSCOM.DLL" (Autor: Burkhard Kainka), nämlich:

```
'Declare Function READBYTE Lib "RSCOM.DLL" () As Integer
  'Liest ein Byte an der geöffneten Schnittstelle (OpenCom) und liefert es als Ergebnis.
  'Es wird eine Zeit lang auf das Zeichen gewartet (der Zeitraum ist durch TIMEOUT zu definieren).
  'Trifft während dieser Zeit kein Zeichen ein, so endet die Funktion und liefert -1 zurück.

'Declare Function TIMEREAD Lib "RSCOM.DLL" () As Double
  'Rückgabe: Zeit in Millisekunden seit dem Aufruf von TIMEINIT.
```

Werden andere RS232-Bibliotheken genutzt, so muss das Lesen der Bytes eventuell abgeändert werden. So gibt es Bibliotheken, die den Versuch zum Lesen eines Zeichens nicht automatisch nach dem Verstreichen einer einstellbaren Zeitspanne abbrechen (Abbruch nach TimeOut). Statt dessen haben sie eine Funktion, mit der man prüfen kann, ob ein Zeichen im Empfangspuffer vorhanden ist. In diesem Fall wird man die Lesefunktion nur dann aufrufen, wenn ein Zeichen im Puffer vorhanden ist, damit das Programm bei unerwartetem Ausbleiben von Zeichen nicht im Leseversuch hängen bleibt. Und die TimeOut-Funktion ist dann auf der Ebene unserer Leseroutine zu realisieren:

```
WHILE NOT (Zeichen_im_Puffer_vorhanden OR TIMER > tOut)
WEND
 'Auslesen oder abbrechen, wenn kein Zeichen nach time out :
IF Zeichen_im_Puffer_vorhanden THEN
  Lese_ein_Byte
ELSE
  EXIT SUB ’Leseprozedur abbrechen
END IF
```

Darin ist `TIMER` die aktuelle Systemzeit des abfragenden Rechners und `tOut` ist der in der Zukunft liegende Zeitpunkt, zu dem spätestens man das letzte Zeichen der Antwort des CLD erwartet (wenige 100 ms nach dem Absenden des Befehls, je nach Baudrate und der möglichen Anzahl der Antwortzeichen). `tOut` ist also vor dem Absenden des Befehls zu berechnen. Falls man einen Systemsekundenzeitzähler befragt, der zur Mitternacht von 86399 auf 0 oder von 86400 auf 1 zurück geht, muss man dies beim Berechnen von `tOut` berücksichtigen:

tOut := Systemsekundenzeit_vor_dem_Absenden_des_Befehls + Antwortzeitspanne
Falls tOut > 86399, dann ist
tOut := tOut - 86399

Leseroutine für die Antwort des CLD

```
Public Function EPRead(CLDAck As Integer, FCByte As Integer, _
                  ThirdByte As Integer, _
                  dat() As String * 17, datz As Integer, BCCTest As Integer, _
                  Vorlauf As Integer) As Long
'Die im RS232-Empfangspuffer eingegangenen Zeichen prüfen,
'dem Beginn des Protokollrahmens voraus gehende Zeichen zählen,
'Daten in das String-Array dat$() sortieren,
'Telegrammstruktur prüfen und Statusvariablen entsprechend setzen.
'Abbruchkriterium: TimeOut oder Ack/Nak..Stx... ETx und gegebenenfalls das BCC gefunden.

'Merkmale:
'Es wird nicht eine bestimmte Anzahl an Zeichen erwartet, sondern nach dem Auftreten von 'ETX' gesucht.
'Das Programm bleibt an fehlerhaften Telegrammen nicht hängen.

Variablen:
'CLDAck: (Output):= Ack (Ack gefunden),
            'Nak (Nak gefunden),
            'KeinEinziges (= kein einziges Zeichen gefunden)
```

```
                'oder keine der drei Konstanten (= Irregular)
'FCByte: (Output):= FehlercodeByte des Antworttelegramms
                'oder Kein (kein Zeichen nach ack/nak gefunden)
'ThirdByte (Output):= ETx, STX oder Fehlt
'dat$(): (Output) : Eindimensionales Stringarray für Daten des Telegramms.
                'Der Inhalt dieser Elemente kann später mit der Funktion VAL() in das Fließkommaformat
                'umgewandelt werden. Es ist nicht sinnvoll, dies von vorneherein zu tun, denn Datenelemente
                ' können alphanumerische Zeichen enthalten, nämlich generell beim Befehl RV und
                'das Zeichen '*' beim Befehl RD der CLD 8xx Serie, wenn keine Daten vorliegen).
                'Der Indexwert zählt ab Null, maximal 12 Elemente werden benötigt.
                'dat$() ist als String-Array mit Elementen fixer Länge (zum Beispiel 'As String * 17')
                'oder als String-Array mit Elementen variabler Länge ('As String') zu deklarieren. Falls
                'der Typ 'As String' verwendet wird, so muss dat$() außerhalb dieser Subroutine,
                'also vor dem Aufrufen dieser Routine, mit einer ausreichenden Anzahl an Zeichen
                'initialisiert werden. Bei zyklischer Anwendung der Routine muss das vor jedem Aufruf
                'erneut geschehen.
                'Die RV-Antwort kann erfahrungsgemäß bis zu 15 Zeichen liefern.
                'Achtung: Wenn dat() mit fixer Stringlänge deklariert worden ist, so werden alle Elemente
                'des Arrays nach rechts mit Leerzeichen aufgefüllt. Bevor diese Arrayelemente in String-
                'vergleichen eingesetzt werden können, müssen die rechts anhängenden Leerzeichen mit
                'RTrim$(dat(x)) entfernt werden.

'datz:   (Output):= Indexwert des letzten gültigen Elements in dat$()
                '= -1, falls keine Daten gefunden

'BCCTest: (Output): Ergebnis der BCC-Überprüfung
                '(nur gültig, wenn ThirdByte = stx, andernfalls
                'wurde regulär kein BCC übertragen)
                '= 0, wenn Übertragung der Daten o.k. war
                '<> 0 bei BCC-Fehler

'Vorlauf (Output):= Anzahl der dem ack/nak voran gegangenen Zeichen

'Rückgabewert:=      der Wert des Bitspeichers RecErr (siehe unten).
                '= 0, falls die empfangene Zeichenkette im Protokollrahmen korrekt ist
                'andernfalls sind definierte Bits gesetzt:
                'KeinEinziges, KeinRegulaeres, FehlercodeFehlt, ThirdByteMissing, ThirdByteIrregular,
                'ETxFehlt, BCCFehlt
'RecErr      sammelt die Bits für den Rückgabewert. Unter Umständen können mehrere Bits gesetzt sein.

'Ablauf:
'Regulären Telegrammanfang ('ack'/'nak') suchen
'Vorlauf := true, falls irreguläre Zeichen voran gehen
'CLD-Fehlercode in FCByte separieren
'Nach STX suchen und gegebenenfalls Antwortdaten im Array dat$() separieren:
'Das Programm erwartet, dass die Datenelemente durch STX, Komma und ETX ge-
'trennt sind. Maximal datlmax Zeichen je Datenelement werden eingelesen.
'Gültige Daten sind in den Elementen dat$(0) bis dat$(datz).
'Die Daten sind zweifelhaft, wenn die Prüfsumme Test nicht Null ergibt,
'gegebenenfalls in RecErr das Bit 'BCFehler' setzen (findet aktuell nicht
'statt, statt dessen wird der Wert BCCTest zurückgegeben)
'datz := -1, falls keine Datenfelder gefunden.
'Abbrechen, wenn 'ETX' und eventuell die Prüfsumme gefunden

Dim d As Integer, easc As Integer, z As Integer, datlMax As Integer
Dim datzMax As Integer
Dim RecErr As Long
datzMax = UBound(dat)
datlMax = Len(dat(datzMax)) 'datlMax Zeichen je Datenelement können aufgenommen werden.
datz = -1

 'Die ersten drei Zeichen suchen:
CLDAck = READBYTE
Vorlauf = 0
If CLDAck <> -1 Then 'ein Zeichen empfangen
```

```
  'Regulären Anfang ('ack' oder 'nak') suchen:
While Not (CLDAck = Ack Or CLDAck = Nak Or CLDAck = -1)
  Vorlauf = Vorlauf + 1
  CLDAck = READBYTE 'Erstes holen
Wend
If CLDAck <> -1 Then 'Ack oder Nak gefunden
  BCCTest = CLDAck
  FCByte = READBYTE 'Zweites holen
  If FCByte = -1 Then
    FCByte = Fehlt
    EPRead = (EtxFehlt Or FehlercodeFehlt)
    ReceiveTime = TIMEREAD
    Exit Function
  Else
    BCCTest = BCCTest Xor FCByte
  End If
   'Zweites reguläres Zeichen gefunden
  ThirdByte = READBYTE
  If ThirdByte = Stx Then 'folgen Daten
    BCCTest = BCCTest Xor ThirdByte
  'Datenseparierung:
  'dat$() = Datenspeicher
  'datz zeigt auf das aktuell zu beschreibende Element in dat$(Index).
  'd zeigt auf die jüngst beschriebene Zeichenposition in dat$(datz)
  'Nach dem Lesen eines Kommas wird datz um 1 erhöht für das Lesen des nächsten Datenfeldes.
  'Nach dem Auffinden von ETx wird das Lesen der Daten beendet.

  'hole restliche Zeichen (nach STX) bis ETX :
     easc = READBYTE 'Viertes Zeichen des Telegramms, das erste Zeichen des ersten Datenfelds
     If easc <> -1 Then
       d = 0: datz = 0
       While easc <> Etx And easc <> -1
          'Separierung in Stringarray:
         If easc = Komma And datz < datzMax Then
           dat(datz) = Left$(dat(datz), d)
           datz = datz + 1: d = 0
         ElseIf d < datlMax Then 'Zur Sicherheit: d wird auf die maximale Datenfeldlänge
                                 '(datlmax Zeichen) begrenzt.
           d = d + 1: Mid$(dat(datz), d) = Chr$(easc)
         End If
         BCCTest = BCCTest Xor easc
         easc = READBYTE
       Wend
   'Jetzt ist ETX auf STX folgend gefunden oder easc = -1
       dat(datz) = Left$(dat(datz), d)
   'letztes Zeichen (Block Check Character Byte):
       If easc = Etx Then
          'wurde dieses jüngste Zeichen noch nicht verarbeitet, also:
         BCCTest = BCCTest Xor easc
         'BCC holen:
         easc = READBYTE
         If easc <> -1 Then
           BCCTest = BCCTest Xor easc
         Else
           RecErr = RecErr Or BCCFehlt 'Bitweise addieren
         End If
       End If
     Else
       RecErr = EtxFehlt Or BCCFehlt
     End If
  ElseIf ThirdByte = Etx Then 'handelt es sich um ein Drei-Zeichen-Telegramm
  ElseIf ThirdByte = -1 Then 'handelt es sich um ein unvollständiges Telegramm
    ThirdByte = Fehlt
    RecErr = ThirdByteMissing Or EtxFehlt
  Else 'war das dritte Zeichen falsch
    RecErr = ThirdByteIrregular
    'Daraufhin folgende Bytes werden nicht ausgelesen
```

```
    End If
  Else
    RecErr = KeinRegulaeres
    CLDAck = Fehlt
    FCByte = Fehlt
    ThirdByte = Fehlt
  End If
Else
  RecErr = KeinEinziges
  CLDAck = Fehlt
  FCByte = Fehlt
  ThirdByte = Fehlt
End If
EPRead = RecErr
ReceiveTime = TIMEREAD
End Function
```