

Application Note 11

Serielle Schnittstellen RS-232 programmieren mit eigerScript Methoden und Beispiele

Erstellt: Januar 2008
Update: 24. Januar 2010

Christoph Angst

© S-TEC electronics AG, CH-6300 Zug
eiger@s-tec.ch
www.s-tec.ch
www.eigergraphics.com



Dieses Dokument können Sie von der Internetseite www.eigergraphics.com als PDF-Datei herunterladen.

Inhaltsverzeichnis

1	Einleitung	3
2	Schnittstellen des eigerPanels	4
2.1	Übersicht über die Anschlüsse des eigerPanels	4
3	RS485 Schnittstelle (COM0)	5
4	RS232 Schnittstellen (COM1 und COM2)	5
4.1	COM1	5
4.1.1	Schema und Stecker von COM1	5
4.1.2	COM1 als Programmier-Schnittstelle	6
4.1.3	COM1 als Debug-Schnittstelle.....	6
4.2	COM2	6
4.2.1	Schema und Stecker von COM2	6
4.2.2	COM2 als Kommunikations-Schnittstelle	7
4.3	Verbindungskabel im StarterKit.....	7
4.4	Prinzip der asynchronen Datenübertragung.....	8
4.5	Datenannahme mit dem Serial Server des eigerPanels	8
5	eigerScript-Methoden der Klasse „Serial“	9
5.1	Baudrate festlegen.....	9
5.2	Daten empfangen	9
5.3	Daten senden	10
5.4	Binäre Daten senden und empfangen mit YMODEM	13
6	Beispiel-Programm „SERI“ für die Kommunikation zwischen PC und eigerPanel.....	15
6.1	ScreenShot der Programm-View SERI_001.EVS	15
6.2	Beschreibung des Beispiel-Programms SERI	16
6.3	Script der Beispiel-Anwendung SERI.....	17
7	Anhang	25
7.1	Anleitung zum Einrichten des Hyper Terminal von Windows XP	25
7.2	ASCII-Zeichensatz	27



1 Einleitung

In dieser Application Note wird gezeigt, welche eigerScript Methoden für den Einsatz der seriellen Schnittstellen RS232 des eigerPanels zur Verfügung stehen. Teil dieser Application Note ist das Programmbeispiel „SERI“. Dieses finden Sie in der eigerDemoCD unter dem Verzeichnis Application Notes oder auf der Download-Seite von www.eigergraphics.com. Sie können den Programmordner „SERI“ direkt auf die CompactFlash™ Card speichern und auf dem eigerPanel 57 ausführen. Beachten Sie dabei, dass die Verzeichnis-Struktur stimmt (inkl. obligatorische Startdatei START.FOX). Eine genaue Anleitung dafür finden Sie im „eigerScript-Schnelleinstieg“ (ebenfalls auf der CD oder auf der Download-Seite von www.eigergraphics.com).

Die Ausführungen in dieser Application Note gelten für alle eigerPanels mit Firmwarestand ab V1.00 (01.10.2009, ersichtlich beim Aufstarten des eigerPanels auf dem Startbild).

2 Schnittstellen des eigerPanels

2.1 Übersicht über die Anschlüsse des eigerPanels

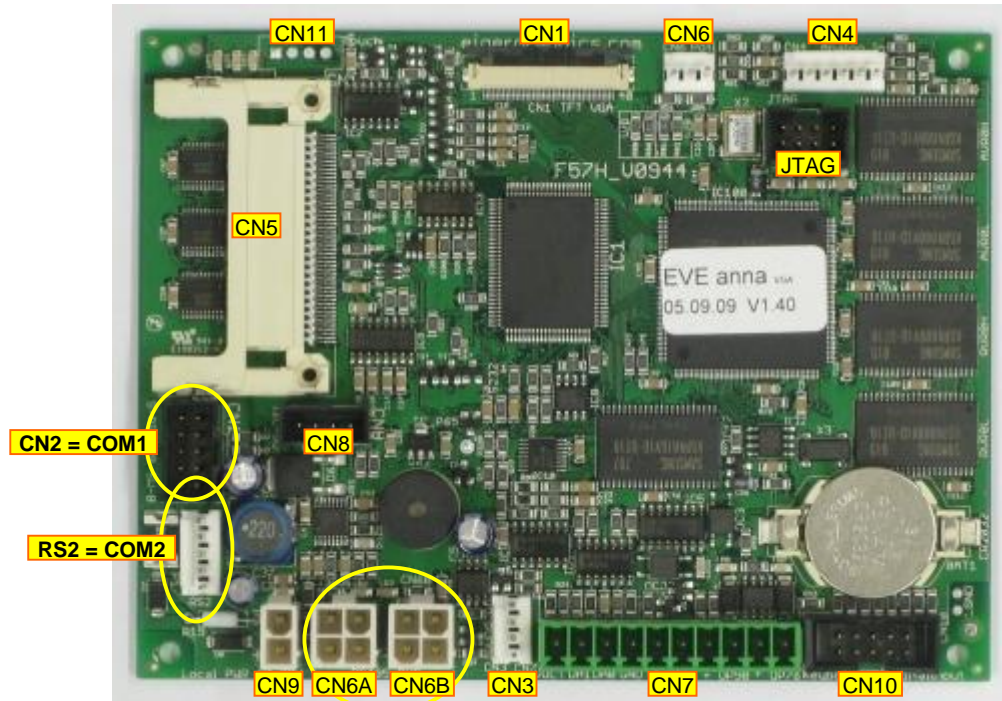


Abbildung 1: Komponenten des embedded Computers F57H_V0944 für das eigerPanel 57

Tabelle 1: Anschlüsse der FOX-Platine (Legende zu Abbildung 1)

Platine F57H_V0944 vom eigerPanel57H

(gilt auch für die Platinen F57C_V0944 und F70H_V1001)

CN1 Anschluss Display TFT VGA mit Backlight und Touch
> OP72: Analog OUT für Display-Backlight PWM (0..1200)

CN2 S-PROG Download / Debug (FOX-COM1 - UART1) – reserviert als Programmier-Schnittstelle für Microprozessor (S-PROG10) und Debugschnittstelle (Adapterkabel-Nr. F4337)

CN3 Analog Ausgänge Sound DA 5.0 V

CN4 Analog Eingänge für NTC, Potentiometer:
PA2: Analog IN 3.3 V (optional bis 10 V)
PA3: Analog IN 3.3 V (optional bis 10 V)

CN5 Schnittstelle Compact Flash Card

CN6 Analog Eingang Potentiometer 3.3 V

CN6A / CN6B BUS, serielle Schnittstelle RS485 12.0V. Kann auch als RS232 Schnittstelle verwendet werden (FOX-COM0)

CN7 Externe Ein-/Ausgänge (Steckertyp: PTR Buchsenleiste mit Schraubanschlüssen):

OP90: Analog OUT 5.0 V, galvanisch getrennt
IP91: Digital IN 3.3 V
VOLT_IN: Analog IN AN1 0..50V (0..1023)
OP76: Analog OUT PWM gesteuert (0..3000) open collector
DA0: Analog OUT 0..255 (0..10VDC)
DA1: Analog OUT 0..255 (0..10VDC)

CN8 Anschluss I²C-BUS (Extern) 5V

CN9 Power Supply VDC 8-30VDC

CN10 Digital Eingänge für bis zu 8 Funktionstasten (Keyboard) 3.3V

CN11 Anschluss Touchscreen (Extern) optional

Buzzer: OP92, aber exkl. für Buzzer ohne parallelen Analog Out

JTAG Programmier-Schnittstelle für EVE anna

RS2 FOX-COM2, serielle Schnittstelle RS232, UART2 zur freien Verwendung (Adapterkabel-Nr. F4259 und F4339)

3 RS485 Schnittstelle (COM0)

Die eigerScript-Befehle sind auch für die RS485-Schnittstelle anwendbar. Bei solcher Verwendung ist diese Schnittstelle mit COM0 anzusprechen.

4 RS232 Schnittstellen (COM1 und COM2)

Das eigerPanel57 und das eigerPanel70 verfügen über 2 serielle Schnittstellen RS232: COM1 (UART 1) und COM2 oder (UART 2) (vgl. Abbildung 1). Diese sind bidirektional. Für den Ground, das Senden und das Empfangen von Daten steht je ein PIN zur Verfügung (Abbildung 2).

Über COM1 läuft der Download und das System Debugging. COM2 ist für Anwendungen frei verfügbar.

4.1 COM1

4.1.1 Schema und Stecker von COM1

Die serielle Schnittstelle RS232 COM1 bezieht sich auf den Stecker CN2 auf der Rechnerplatine (Abbildung 1).

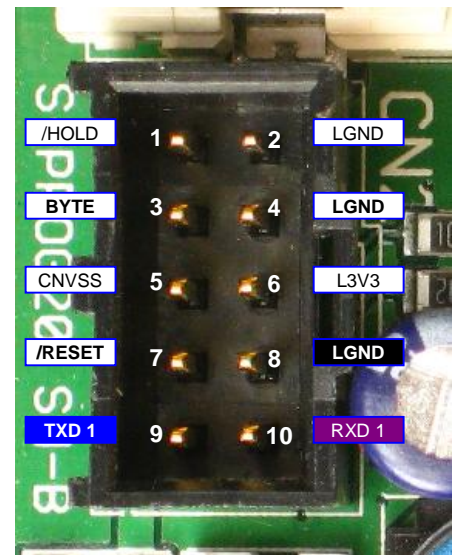
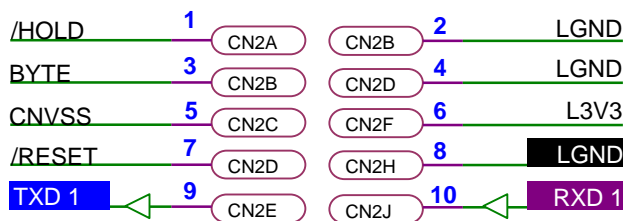


Abbildung 2: Serielle Schnittstelle COM1 (UART 1) auf CN2 des eigerPanels. Die Methoden der Class Serial von eigerScript beziehen sich auf die PINs 8 (für GROUND), 9 (für Senden) und 10 (für Empfangen). Über die PINs 1,3,5 läuft der M16-Download.



Farben für die Verbindungskabel:

Senden TXD1: **blau**

Empfangen RXD1: **violett**

Masse LGND: **schwarz**

Diese Farbkonvention ist eine Empfehlung, an welche wir uns auch bei den Verbindungskabeln halten, welche dem StarterKit des eigerPanels beigelegt sind (vgl. Tabelle 2, S.7).

4.1.2 COM1 als Programmier-Schnittstelle

Über die serielle Schnittstelle wird beispielsweise die Firmware aktualisiert (Upgrade). Als Upgrade-Schnittstelle des Betriebssystems steht COM1 jedoch für den Nutzer nicht im Vordergrund, da dies auf einfachere Weise über die CompactFlash™ Card möglich ist. Die neueste Firmware ist jeweils auf der Download-Seite von www.eigergraphics.com aufgeschaltet, inkl. Anleitung (ReadMe).

4.1.3 COM1 als Debug-Schnittstelle

Beim Entwickeln von Anwendungen für das eigerPanel ist COM1 vor allem als Debug-Schnittstelle vorgesehen. Mit den Methoden der Klasse „**Debug**“ lassen sich Texte und Variablenwerte darüber ausgeben. Über die Fehlersuche bei der Software-Entwicklung gibt es eine spezielle Application-Note: „22 Debug – Fehlern auf der Spur“.

4.2 COM2

4.2.1 Schema und Stecker von COM2

Die serielle Schnittstelle RS232 COM2 bezieht sich auf den Stecker RS2 auf der Rechnerplatine (Abbildung 3).

RS232-Schnittstelle RS2
CST 100 5 pol

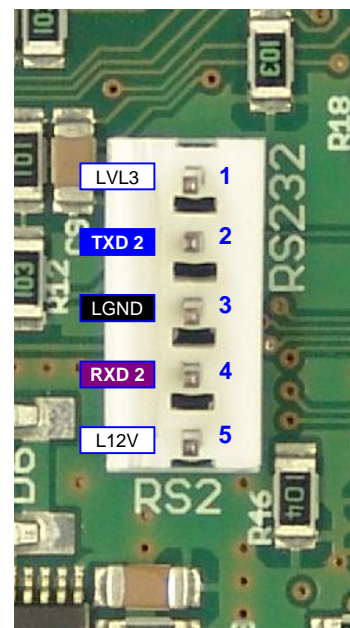
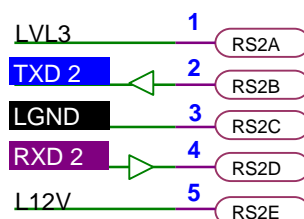


Abbildung 3: Schema und Steckleiste der Schnittstelle COM2 (UART2).

4.2.2 COM2 als Kommunikations-Schnittstelle

Die serielle Schnittstelle RS232 COM2 ist für Ihre Anwendungen frei verfügbar. Die folgenden Ausführungen sind dieser Schnittstelle gewidmet. Auch die dabei vorgestellten eigerScript Methoden der **Klasse „Serial“** beziehen sich auf diese Schnittstelle.

4.3 Verbindungskabel im StarterKit

Dem StarterKit sind drei verschiedene Verbindungskabel für die beiden seriellen Schnittstellen beigelegt (vgl. Tabelle 2).

Tabelle 2: Kabel für Serielle Schnittstellen im eigerDemoKit

	<p>Kabel FOX-COM1 (UART1) mit Reset</p>	<p>Das FOX-COM1 Kabel (UART1) ist eine serielle Schnittstelle für das Debugging. Der Reset-Taster dient zum Neustart des FOX57.</p>
	<p>Kabel FOX-COM2 (UART2) (female)</p>	<p>Das FOX-COM2 Kabel (UART2) ist eine serielle Schnittstelle „RS232“ (→ weiblicher SUBD9 Stecker)</p>
	<p>Kabel FOX-COM2 als DTE</p>	<p>Das FOX-COM2 Kabel (UART2) ist eine serielle Schnittstelle RS232 (→ männlicher SUBD9 Stecker)</p>

4.4 Prinzip der asynchronen Datenübertragung

Das Senden und Empfangen über die serielle Schnittstelle RS-232 (EIA-232) des eigerPanels funktioniert nach dem Prinzip der „asynchronen Datenübertragung“. Die Übertragung kann zu einer beliebigen Zeit einsetzen und wird durch ein sogenanntes Start-Bit eingeleitet (Abbildung 4). Darauf folgen die 8-Bits des Zeichens, das übertragen wird. Die Übermittlung des Zeichens wird mit einem Stopp-Bit abgeschlossen.

Fürs eigerPanel gilt:

- Stoppbit: 1
- Datenbits: 8
- Parität: keine

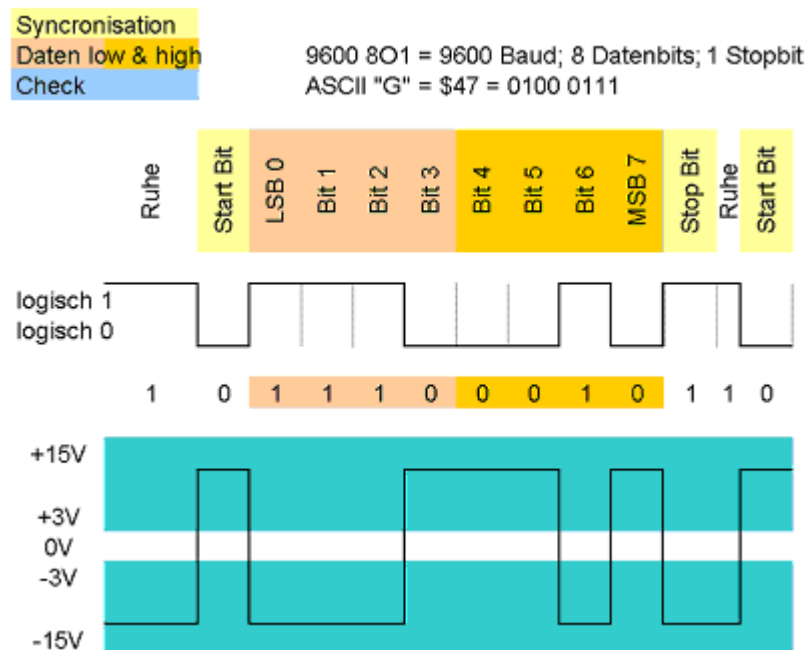


Abbildung 4: Asynchroner Datenstrom für den Buchstaben G über die serielle Schnittstelle RS-232. Im unteren Bereich ist der physikalische Spannungsverlauf und entsprechend im oberen Bereich der logische Signalverlauf dargestellt (Abbildung aus <http://de.wikipedia.org/wiki/UART>, leicht verändert).

4.5 Datenannahme mit dem Serial Server des eigerPanels

Der **Serial Server** bedient die seriellen Schnittstellen. Er hat die Aufgabe, asynchron zum Programmablauf eintreffende Zeichen entgegenzunehmen und zwischenspeichern. Die Zeichen gelangen nach dem Empfang durch die UART in einen **RingBuffer**, der im Falle von COM2 maximal 1024 Zeichen zwischenspeichern kann. Mit Hilfe der wiederholten Verwendung der eigerScript Methode „Serial.Rx_Char()“ wird der Reihe nach Zeichen um Zeichen aus dem RingBuffer herausgeholt. Ist der RingBuffer leer, dann liefert die Methode Serial.Rx_Char() den Wert „-1“.

5 eigerScript-Methoden der Klasse „Serial“

Die Methoden für das Management der seriellen Schnittstellen RS232 und RS485 sind in eigerScript in der Klasse „Serial“ zusammengefasst. Die nachfolgende Beschreibung dieser Klasse und der zugehörigen Methoden sind zum Teil dem Befehlskatalog „Befehlsreferenz_eigerScript_eVM.pdf“ entnommen.

5.1 Baudrate festlegen

Serial.SetBaudrate(VarInt:COMx,VarInt:Baud_9600)

Mit dem Befehl `Serial.SetBaudrate(VarInt:COM1,VarInt:Baud_9600)` wird die Bitrate für die Datenübertragung bestimmt, z.B. 9600 bit/s. In eigerScript stehen 9 Bitraten als Konstanten zur Auswahl (vgl. Tabelle 3). Die programmierte Baudrate muss mit der Baudrate des korrespondierenden Gerätes übereinstimmen.

Tabelle 3: Vordefinierte Bitraten in eigerScript.

Bitrate (bit/s)	Syntax
1'200	Baud_1200
2'400	Baud_2400
4'800	Baud_4800
9'600	Baud_9600
19'200	Baud_19200
38'400	Baud_38400
57'600	Baud_57600
115'200	Baud_115200
250'000	Baud_250000

Beispiel-Code 1:

```
Serial.SetBaudrate ( COM2 , Baud_38400 )
```

→ vgl. auch
Beispiel-Code 8,
S.19, Zeilennr.
188

5.2 Daten empfangen

Serial.Rx_Char(VarInt:COMx,VarInt:Char)

Die Zeichen gelangen nach dem Empfang durch die UART in einen RingBuffer, der im Falle von COM2 maximal 1024 Zeichen zwischenspeichern kann. Mit Hilfe der eigerScript Methode „Serial.Rx_Char()“ wird der Reihe nach Zeichen um Zeichen aus dem RingBuffer herausgeholt. Die Methode kopiert das Zeichen in eine vordeklarierte Integer-Variable (vgl. Beispiel-Code 2). Der Befehl empfängt Werte von 0 bis 255 (gemäss Kodierung des ASCII-Zeichensatzes, vgl. Tabelle

5, S.28). Falls gerade kein Zeichen ansteht, erhält die Integer-Variable den Wert „-1“.

COM_x kann sein COM0 (RS485 Schnittstelle) oder COM2 (RS232 Schnittstelle).

Beispiel-Code 2: Das im Ringbuffer von COM1 nächste Zeichen wird in die Variable `InChar.I` kopiert, um z.B. in einer IF-Schleife weiter verwendet zu werden.

→ vgl. auch
Beispiel-Code 8,
S.21, Zeilenr.
367

```
Serial.Rx_Char ( COM2, InChar.I )
IF InChar.I == "?" THEN
  CallSubroutine (GoToView1)
ELSIF InChar.I == "!" THEN
  ...
```

Beispiel-Code 3: Mit Hilfe eines Timers wird der Ringbuffer für COM2 periodisch auf eingegangene Zeichen überprüft. Die LOOP..ENDLOOP-Schleife sorgt dafür, dass die empfangenen Zeichen mit der Methode `Serial.Rx_Char(COM2, Char.I)` der Reihe nach aus dem Ringbuffer in die Integer-Variable `Char.I` verschoben werden, bis der Ringbuffer wieder leer ist und mit dem Wert „-1“ meldet, dass kein Zeichen mehr vorhanden ist. Das Beispiel stammt aus der Anwendung „SERI“ (Beispiel-Code 8).

```
SUB Timer4COM
  Timer.InstallLocal(0,Read_COM)
  Timer.Load(0, 10)
  Timer.StartContinuous(0)
ENDSUB

SUB Read_COM
  Serial.Rx_Char(COM2, Char.I)

  LOOP

  IF Char.I != -1 THEN           ; -1 means empty Buffer
    Str.AddChar(InputString.$,Char.I)
    Str.Length(StringLength.I, InputString.$)

    IF StringLength.I >= MaxStringLength.I THEN
      Str.RemoveChar_at_Position(eI.Garbage, InputString.$, 1)
    ENDIF

    CallSubroutine(Draw_InputField)
  ELSE
    EXITLOOP
  ENDIF

  ENDLLOOP

ENDSUB
```

5.3 Daten senden

Serial.Tx_Char(VarInt:COMx,VarInt:Char)

Mit der Methode `Serial.Tx_Char(VarInt:COMx,VarInt:Char)` wird ein einzelnes Zeichen über die Serielle Schnittstelle gesendet. Das Zeichen kann direkt der Methode übergeben werden oder der vordefinierte Inhalt einer Integer-variablen sein.

COM_x kann sein COM0 (RS485 Schnittstelle) oder COM2 (RS232 Schnittstelle).

Beispiel-Code 4: Verschiedene Möglichkeiten, das Zeichen „3“ zu senden – als Inhalt der zuvor deklarierten Integervariable Char.I, als Ziffer 3 in Anführungszeichen, als ASCII-Wert des Zeichens (3x16+3=51) oder als ASCII-Wert des Zeichens in hexadezimaler Schreibweise (vgl. Tabelle 5, S.28).

→ vgl. auch
Beispiel-Code 8,
S.21, Zeilennr.
321

```
Char.I := "3" ; oder: 51; oder: 0x33
Serial.Tx_Char(COM2, Char.I) ; oder: "3"; oder: 51; oder: 0x33
```

oder

```
Serial.Tx_Char(COM2, "3" )
```

oder

```
Serial.Tx_Char(COM2, 51 )
```

oder

```
Serial.Tx_Char(COM2, 0x33 )
```

Serial.Tx_CRLF(VarInt:COMx)

Mit dieser Methode wird ein CRLF (Zeilenumbruch und Zeilenvorschub, *engl.* Carriage Return und Line Feed) über die serielle Schnittstelle ausgegeben, um eine Zeile abzuschliessen. Der Hex-Wert von CRLF ist 0x0D (CR) und 0x0A (LF), vgl. Tabelle 6, S.28).

COMx kann sein COM0 (RS485 Schnittstelle) oder COM2 (RS232 Schnittstelle).

Beispiel-Code 5: Sendet einen Zeilenumbruch (CRLF oder CR) über die Schnittstelle COM2.

→ vgl. auch
Beispiel-Code
8, S.20,
Zeilennr. 232

```
Serial.Tx_CRLF(COM2)
```

Serial.Tx_NUL(VarInt:COMx)

Mit dem zuvor beschriebenen Befehl `Serial.Tx_Char(COMx,Char)` können alle Zeichen gesendet werden, auch das Zeichen NUL. Gewisse Anwendungen bedienen sich des NUL-Zeichens als spezielles Steuerzeichen. In diesen Fällen kann auch der Befehl `Serial.Tx_NUL(VarInt:COMx)` als etwas kürzere Schreibweise benutzt werden. Der Hex-Wert von NUL ist 0x00 (vgl. Tabelle 6, S.28)

COMx kann sein COM0 (RS485 Schnittstelle) oder COM2 (RS232 Schnittstelle).

Beispiel-Code 6: Sendet das Zeichen NUL (0x00)

→ vgl. auch
Beispiel-Code
8, S.20,
Zeilennr. 255

```
Serial.Tx_NUL(COM2)
```

Serial.Tx_String(VarInt:COMx,VarStr:String)

Mit dieser Methode wird ein String über die serielle Schnittstelle ausgegeben. Der Nullterminierte String wird ohne die abschliessende Null gesendet. Muss

eine Null gesendet werden, wird der Befehl `Serial.Tx_NUL(VarInt)` verwendet.

COM_x kann sein COM0 (RS485 Schnittstelle) oder COM2 (RS232 Schnittstelle).

Beispiel-Code 7: Sendet den Inhalt der Stringvariable `Text.$` über die Schnittstelle COM2. Statt eines Variablennamens kann auch direkt eine Zeichenfolge in Hochkommas angegeben werden, z.B. 'Hallo'.

→ vgl. auch
Beispiel-Code
8, S.21,
Zeilenr. 297

```
Serial.Tx_String(COM2,Text.$)
```

oder

```
Serial.Tx_String(COM2, 'Hallo')
```

5.4 Binäre Daten senden und empfangen mit YMODEM

Serial.SendFromFile(VarInt:COMx,VarStr:Filename)

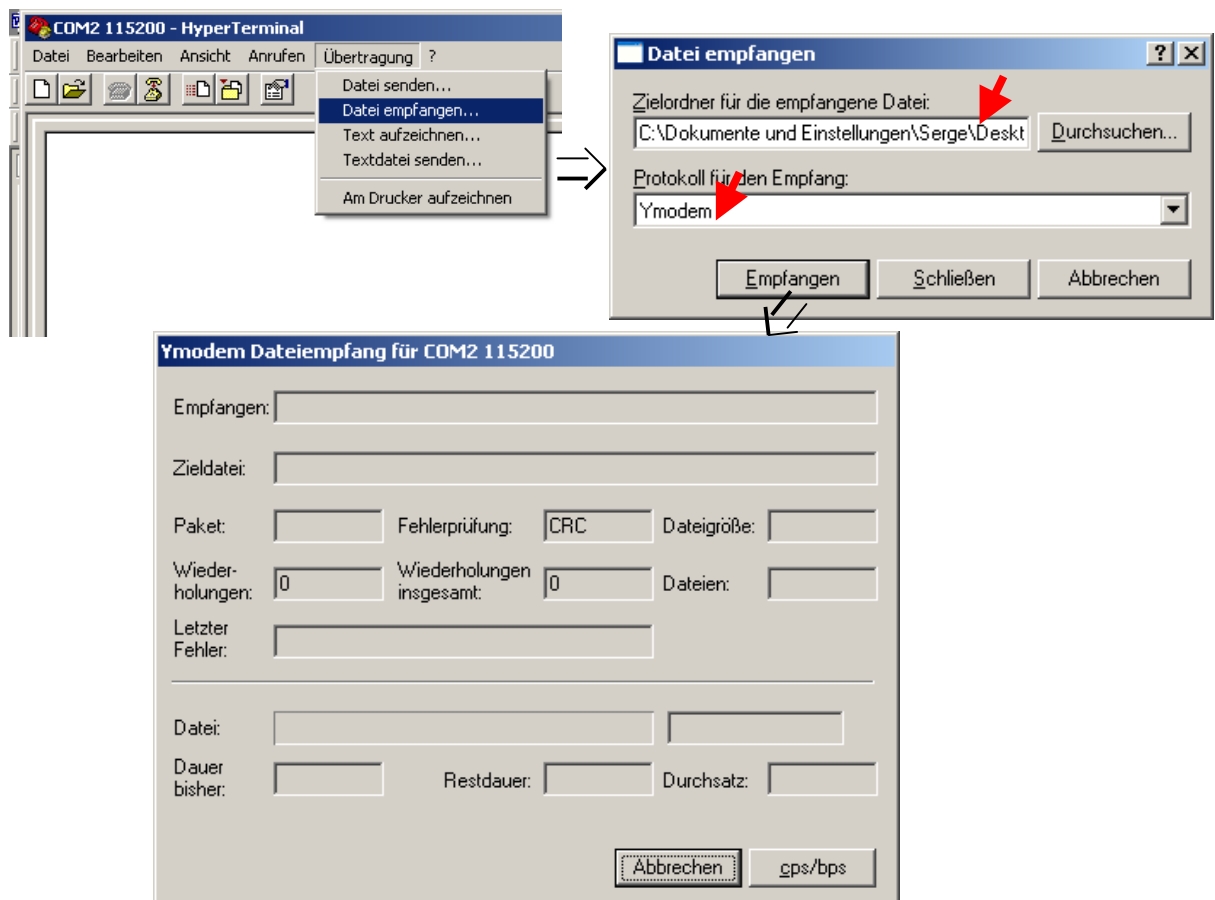
Mit dieser Methode werden beliebige Dateien direkt von der CompactFlash Card an eine serielle Schnittstelle ausgegeben.

COMx: kann sein COM0 (RS485 Schnittstelle) oder COM2 (RS232 Schnittstelle).

Filename: Dateiname mit Pfad. Beispiel: 'C:/TG12/PICT/MyPict.egi'.

Das verwendete Protokoll ist YMODEM mit CRC Prüfsumme. Dadurch lässt sich die Datei mit einem PC und dem Kommunikationsprogramm ‚HyperTerminal‘ (im Windows-menu: Start->Zubehör->Kommunikation) empfangen. Zu beachten ist, dass im HyperTerminal ein Empfangsordner anzugeben ist, in den die Datei unter dem reinen Dateinamen, in unserem Beispiel MyPict.egi, gespeichert wird.

Der Befehl `Serial.SendFromFile(VarInt: COMx, VarStr: Filename)` sollte in eigerScript aufgerufen werden bevor im HyperTerminal „Datei empfangen“ angewählt wird.

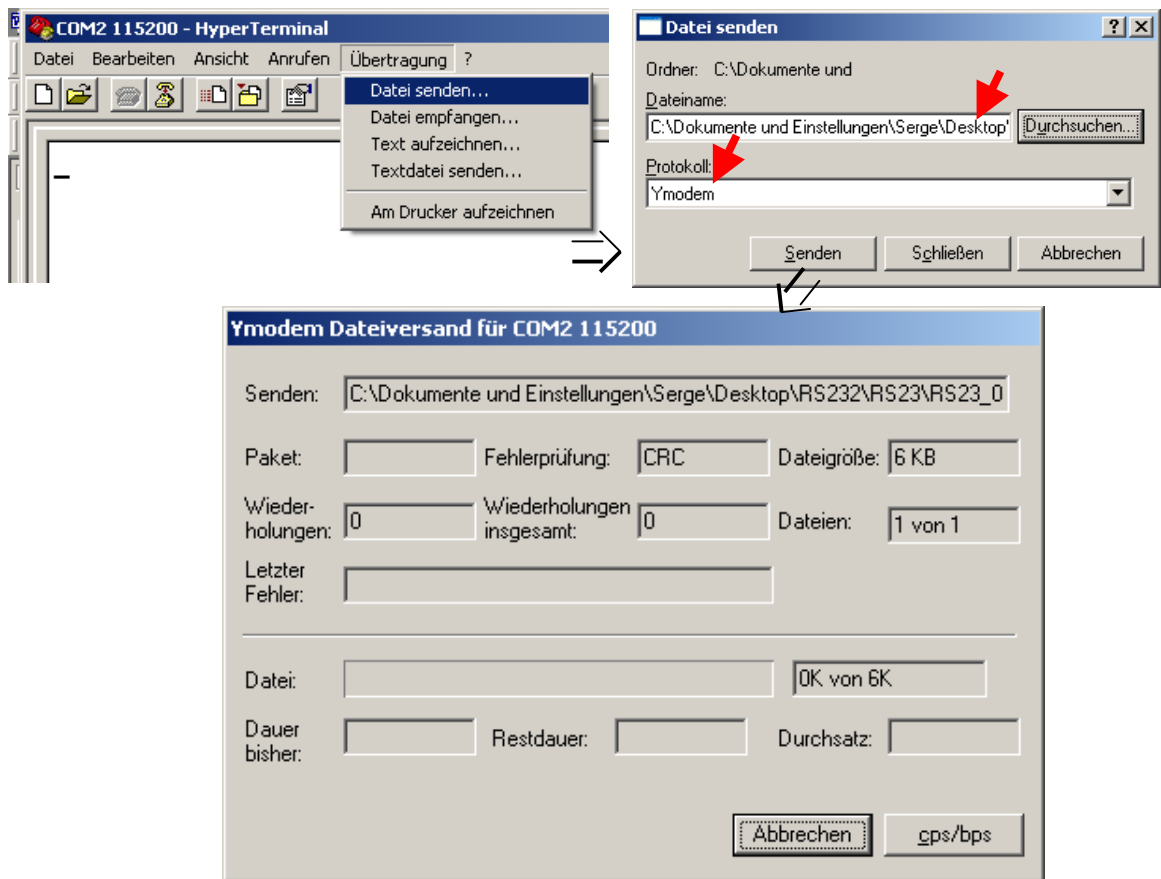


Serial.ReceiveToFile(VarInt:COMx,VarStr:Zielordner)

Eine beliebige mit dem YMODEM-Protokoll übertragene Datei lässt sich mit dem eigerScript-Befehl `Serial.ReceiveToFile(..)` empfangen und direkt in eine Datei auf dem CompactFlash speichern.

COMx: Kann sein COM0 für die RS485-Schnittstelle oder COM2 für die RS232-Schnittstelle.

Zielordner: Zielordner in den die Datei gespeichert werden soll, falls die Datei mit einem Kommunikationsprogramm wie ‚HyperTerminal‘ übermittelt wird. Beispiel: `'C:/TG12/PICT/'`. Der Zielordner muss mit einem `/'` abgeschlossen werden. Nach dem Empfang der Datei steht in der Variablen `Zielordner` der komplette Pfad mit Dateinamen der empfangenen Datei. Die Variable sollte deshalb nicht durch eine konstanten String ersetzt werden. Falls die Datei mithilfe von `Serial.SendFromFile(..)` von einem anderen eigerPanel übermittelt wird, so wird die Datei in den exakt gleichen Ordner wie die übermittelte Quelldatei gespeichert. In diesem Fall kann man für das Argument `Zielordner` einen leeren String angeben (`''`).

**Mögliche Fehlerquellen:**

- **Baudrate:** Die Baudraten müssen natürlich auf Sender und Empfängerseite gleich sein. Dafür steht der Befehl `Serial.SetBaudrate(..)` zur Verfügung. Weitere Einstellungen vgl. Abbildung 8, S.26.

- *Filename*: Der Filename darf höchstens 8 Zeichen sowie 3 Erweiterungszeichen umfassen, z.B. YourName.txt.
- *Verzeichnisstruktur*: Wenn Sie vom PC eine Datei aufs Panel laden wollen, müssen die Verzeichnisstruktur und der Zielordner, die Sie im Befehl „Befehl `Serial.ReceiveToFile(...)` angeben, auf der CFC bereits existieren.
- *Verbindungskabel*: Verwenden Sie ein Nullmodemkabel. Das Nullmodem kreuzt RxTx-TxRx. Beim Seriellen Kabel F4339, welches dem StarterKit beiliegt, ist die Kreuzung bereits vorhanden. Wenn Sie dieses zwischenschalten, funktioniert die Verbindung auch mit einem normalen seriellen Verbindungskabel.

6 Beispiel-Programm „SERI“ für die Kommunikation zwischen PC und eigerPanel

6.1 ScreenShot der Programm-View SERI_001.EVS

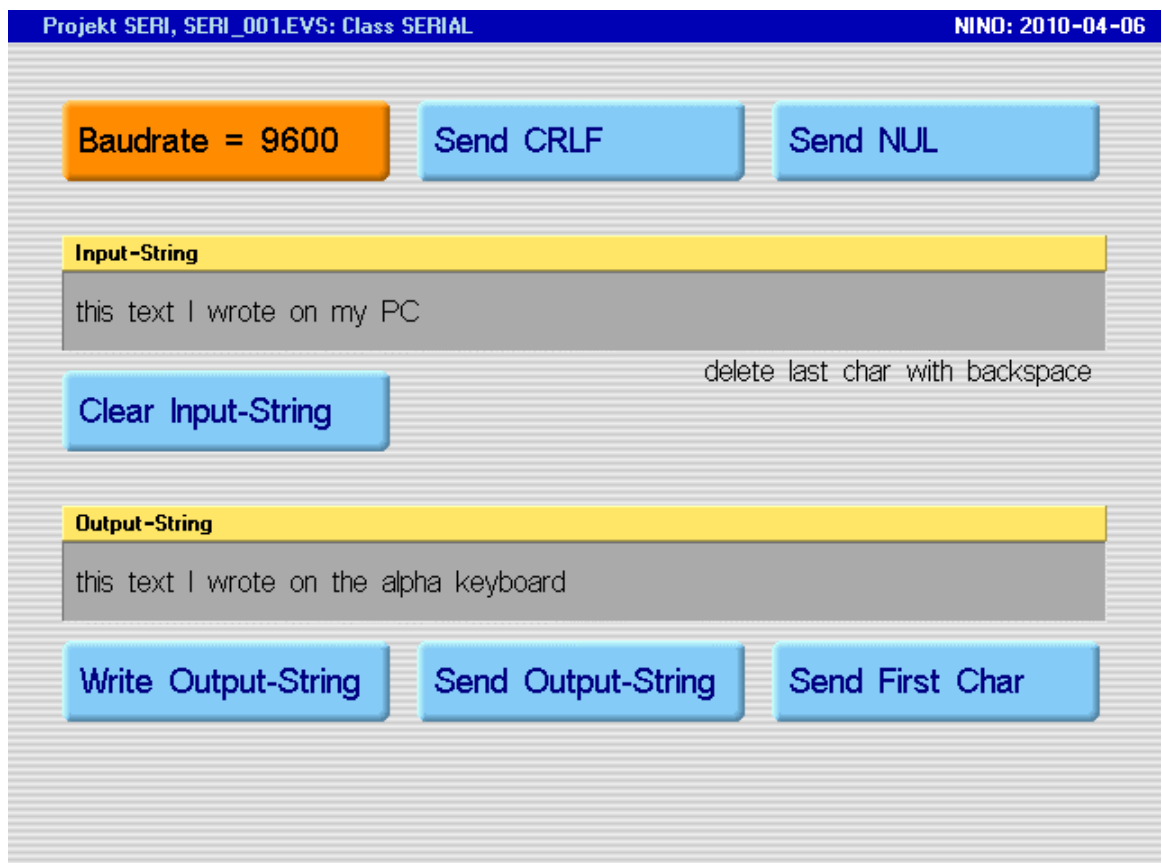


Abbildung 5: ScreenShot der View des Beispiel-Programms SERI. Beim Betätigen des Buttons „Write Output-String“ erscheint das alpha keyboard-Control, womit Sie den Text des Output-Strings eingeben können.

6.2 Beschreibung des Beispiel-Programms SERI

Im Programm-Script „SERI“ finden Sie alle Befehle der Serial-Class angewendet.

Mit SERI können Sie auf dem eigerPanel über die Schnittstelle COM2 Zeichen vom PC empfangen bzw. zum PC senden.

Das Programm beobachtet, welche Zeichen vom PC über die serielle Schnittstelle COM2 an den RingBuffer des eigerPanels gesendet werden, bzw. liest diese – getrieben durch einen Timer – ständig aus dem RingBuffer in einen Input-String ein. Zeitnah mit dem Eingang eines Zeichens in den RingBuffer wird dieses dann als Teil des String-Inhalts auf dem Bildschirm im Textfeld „Input-String“ angezeigt. (vgl. Abbildung 5). Im Beispiel-Programm ist für den InputString eine maximale Anzahl Zeichen festgelegt, damit der Text nicht über das Textfeld „hinausfließt“. Wird die maximale Anzahl Zeichen erreicht, so wird der Text zum Fliesstext.

Das Programm ermöglicht das Löschen des Input-Strings mit Hilfe des Buttons „Clear Input-String“.

Mit der Taste BackSpace Ihres PC's können Sie das letzte Zeichen des Input-Strings im eigerPanel löschen. Der Hex-Wert für BackSpace ist 0x08 (vgl. Tabelle 6, S.28).

Die Beispiel-Anwendung enthält ein weiteres Textfeld für den Output-String, dessen Inhalt Sie mit dem Alpha Keyboard-Control des eigerPanels eingeben können, wenn Sie den Button „Write Output-String“ drücken. Mittels des Buttons „Send Output-String“ senden Sie den ganzen Text im OutputString über COM2 beispielsweise zu Ihrem PC.

Mit dem Button „Send Char“ können Sie auch Zeichen um Zeichen des Output-Strings an den PC senden.

Mit dem Buttons „Send CRLF“ können Sie das Zeichen für eine neue Zeile senden (CRLF = neue leere Zeile). Entsprechendes geschieht mit dem Button „Send NUL“.

Das Script des Beispiel-Programms SERI finden Sie im nachfolgenden Kapitel. Sie können das ausführbare Programm auch von der Download-Seite unserer Homepage www.eigergraphics.com herunterladen.



Um das Programm zu Testen und die Kommunikation zwischen PC und eigerPanel durchzuspielen, brauchen Sie ein Terminalprogramm. Ein solcher gehört zur Grundausrüstung von Windows. Eine Anleitung zum Einrichten des HyperTerminals von Windows XP finden Sie im Kapitel 7.1, S.25 ff.

6.3 Script der Beispiel-Anwendung SERI

Beispiel-Code 8: vollständiger Programmcode der Beispiels-Anwendung SERI

```


2
4 ; -----
4 ; Titel      : Serial-Class   View 601
4 ; File       : Extrahiertes TG12_601.EVS
6 ; -----
6 ; Compiler   : eigerScript
8 ;
8 ; System     : eigergraphics.com; FOX embedded computer
10 ;
10 ; Beschreibung: Demo-Anwendung für die Methoden der Serial-Class
12 ;
12 ; Version    : Initialversion: 04.04.2010
14 ;
14 ; Autor      : Christoph Angst, S-TEC electronics AG
16 ;
16 ;
18 ;
18 ; -----
20 ; (c) 2005-2010 S-TEC electronics AG, CH-6300 Zug; 041 754 50 10
20 ;               www.s-tec.ch                               www.eigergrapics.ch
22 ; -----
24
24 EIGERPROJECT 'SERI' ; Projektbezeichnung: erster Teil
26 des EVI-Filemens
26 EIGERVIEW 1 ; Viewnummer: zweiter Teil des EVI-
28 Filenames: SERI_001.EVI
30
30 ; INCLUDEFILE 'TG12_Common.EVS' ; Gemeinsame Definitionen
32
32 INCLUDEFILE 'CTRL_KBD1.EVS' ; Keyboard
34
36
36 Deklarationen → ; D E K L A R A T I O N E N
38
38 ; Declarations for layout of buttons and fields -----
40
40 CONST Frame_X = 30 ; Distance to Display-Border
42 CONST Frame_Y = 50 ; Distance to Display-Border
44
44 CONST ButtonWidth = 182
44 CONST ButtonHeight = 45
46
46 CONST Space_X = 15
46 CONST Space_Y = 30
48
48 CONST ButtonLeft_X = Frame_X
50 CONST ButtonMid_X = ButtonLeft_X + ButtonWidth + Space_X
50 CONST ButtonRight_X = ButtonLeft_X + 2*(ButtonWidth + Space_X)
52
54
54 CONST ButtonRow01_Y = Frame_Y
54 CONST ButtonRow02_Y = ButtonRow01_Y + ButtonHeight + Space_Y
56 CONST ButtonRow03_Y = ButtonRow02_Y + ButtonHeight + Space_Y
56 CONST ButtonRow04_Y = ButtonRow03_Y + ButtonHeight + Space_Y
58 CONST ButtonRow05_Y = ButtonRow04_Y + ButtonHeight + Space_Y
58 CONST ButtonRow06_Y = ButtonRow05_Y + ButtonHeight + Space_Y
60
60 INTEGER FieldHeader_Y.I
62 CONST FieldHeaderHeight = 20
62 CONST FieldHeaderWidth = 100
64
64 CONST InputField_X = Frame_X

```

```

66  CONST      InputField_Y = ButtonRow02_Y + FieldHeaderHight
INTEGER      InputFieldWidth.I      ; according to eI.DisplayWidth (VGA or
68  WVGA) > SUB LayoutDef
CONST      InputFieldHight = ButtonHight
70
72  CONST      OutputField_X = InputField_X
CONST      OutputField_Y = ButtonRow04_Y + FieldHeaderHight
74
76  INTEGER      BaudrateFlag.I = 4
INTEGER      BAUDRATE.I
78  STRING [20]  Baudrate.$
STRING [30]  FieldHeader.$ = ''
80
82  INTEGER      Char.I
INTEGER      StringLength.I      ; Number of chars in InputString
CONST      MaxStringLengthVGA = 60      ; maximal length of InputString for
84  VGA Display
CONST      MaxStringLengthWVGA = 90      ; maximal length of InputString for
WVGA Display
86  INTEGER      MaxStringLength.I      ; maximal length of InputString
STRING      [100]  InputString.$ = ''      ; Input from Hyperterminal
88
90  STRING [80]  OutputString.$      ; Output to Hyperterminal
92
94  ; Definitionen für PageLayout -----
STRING      [60]  Titel = 'Projekt SERI, SERI_001.EVS: Class SERIAL'
96  STRING      [60]  Titel_Right = ErstellDatum      ; TitelLeiste rechts
98  CONST      PL01_ViewLeft = 1      ; letzte View links
CONST      PL01_ViewRight = 1      ; letzte View rechts
100  CONST      PL01_HomeView = 1      ; HomeView
102  INCLUDEFILE 'TG12_PageLayout_01.EVS'
104
106  INTEGER      Focus.I = 0      ; Aktueller Fokus
CONST      FocusMax = 21      ; Anzahl Fokusbytes
108
110  ; S U B R O U T I N E N _____
112  ;
114  ; Layout-Routine for Layout-Definitions dependent on the Display (VGA / WVGA)
116  ;
118  SUB Layout_Def
120      InputFieldWidth.I := eI.DisplayWidth - 2 * Frame_X
122      IF eI.DisplayWidth == 640 THEN
124          MaxStringLength.I := MaxStringLengthVGA
126      ELSE
128          MaxStringLength.I := MaxStringLengthWVGA
130      ENDIF
132  ENDSUB
134  ; _____
136  ; Buttons
138  ; _____
140  ; Button to set Baudrate -----
SUB SetBaudrate_Leave
142  Fill.LabelParameter(PL01_FNL_Button_UP_Style)
144  Load.Geometry_XYWH(ButtonLeft_X, ButtonRow01_Y, ButtonWidth, ButtonHight)
146  Label.Color(darkorange)
148  Label.Text(Baudrate.$)
150  ENDSUB

```

Subroutinen 

```

142 SUB SetBaudrate_Down
143     Fill.LabelParameter(PL01_FNL_Button_DN_Style)
144     Load.Geometry_XYWH(ButtonLeft_X, ButtonRow01_Y, ButtonWidth, ButtonHeight)
145     Label.Color(darkkhaki)
146     Label.Text(Baudrate.$)
147 ENDSUB
148
149
150 SUB SetBaudrate_Up
151     CallSubroutine(ChooseBaudrate)
152     CallSubroutine(SetBaudrate_Leave)
153 ENDSUB
154
155
156 SUB ChooseBaudrate
157     IF BaudrateFlag.I == 1 THEN
158         BAUDRATE.I := Baud_1200
159         Baudrate.$ := 'Baudrate = 1200'
160     ELSIF BaudrateFlag.I == 2 THEN
161         BAUDRATE.I := Baud_2400
162         Baudrate.$ := 'Baudrate = 2400'
163     ELSIF BaudrateFlag.I == 3 THEN
164         BAUDRATE.I := Baud_4800
165         Baudrate.$ := 'Baudrate = 4800'
166     ELSIF BaudrateFlag.I == 4 THEN
167         BAUDRATE.I := Baud_9600
168         Baudrate.$ := 'Baudrate = 9600'
169     ELSIF BaudrateFlag.I == 5 THEN
170         BAUDRATE.I := Baud_19200
171         Baudrate.$ := 'Baudrate = 19200'
172     ELSIF BaudrateFlag.I == 6 THEN
173         BAUDRATE.I := Baud_38400
174         Baudrate.$ := 'Baudrate = 38400'
175     ELSIF BaudrateFlag.I == 7 THEN
176         BAUDRATE.I := Baud_57600
177         Baudrate.$ := 'Baudrate = 57600'
178     ELSIF BaudrateFlag.I == 8 THEN
179         BAUDRATE.I := Baud_115200
180         Baudrate.$ := 'Baudrate = 115200'
181     ELSIF BaudrateFlag.I == 9 THEN
182         BAUDRATE.I := Baud_250000
183         Baudrate.$ := 'Baudrate = 250000'
184     BaudrateFlag.I := 0
185     ENDF
186     BaudrateFlag.I := BaudrateFlag.I + 1
187
188     Serial.SetBaudrate(COM2,BAUDRATE.I)
189
190 ENDSUB
191
192 ; Button to clear InputField -----
193
194 SUB ClearInput_Leave
195     Fill.LabelParameter(PL01_FNL_Button_UP_Style)
196     Load.Geometry_XYWH(ButtonLeft_X, ButtonRow03_Y, ButtonWidth, ButtonHeight)
197     Label.Text('Clear Input-String')
198 ENDSUB
199
200
201
202 SUB ClearInput_Down
203     Fill.LabelParameter(PL01_FNL_Button_DN_Style)
204     Load.Geometry_XYWH(ButtonLeft_X, ButtonRow03_Y, ButtonWidth, ButtonHeight)
205     Label.Text('Clear Input-String')
206 ENDSUB
207
208
209 SUB ClearInput_Up
210     CallSubroutine(ClearInput_Leave)
211     Str.Clear(InputString.$)
212     CallSubroutine(Draw_InputField)
213 ENDSUB
214
215 ; Button to send CRLF -----
216
217 SUB SendCRLF_Leave
218     Fill.LabelParameter(PL01_FNL_Button_UP_Style)

```

```
218     Load.Geometry_XYWH(ButtonMid_X, ButtonRow01_Y, ButtonWidth, ButtonHight)
219     Label.Text('Send CRLF')
220 ENDSUB
221
222
223 SUB SendCRLF_Down
224     Fill.LabelParameter(PL01_FNL_Button_DN_Style)
225     Load.Geometry_XYWH(ButtonMid_X, ButtonRow01_Y, ButtonWidth, ButtonHight)
226     Label.Text('Send CRLF')
227 ENDSUB
228
229
230 SUB SendCRLF_Up
231     CallSubroutine(SendCRLF_Leave)
232     Serial.Tx_CRLF(COM2)
233 ENDSUB
234
235
236 ; Button to send NUL -----
237 SUB SendNUL_Leave
238     Fill.LabelParameter(PL01_FNL_Button_UP_Style)
239     Load.Geometry_XYWH(ButtonRight_X, ButtonRow01_Y, ButtonWidth,
240 ButtonHight)
241     Label.Text('Send NUL')
242 ENDSUB
243
244
245 SUB SendNUL_Down
246     Fill.LabelParameter(PL01_FNL_Button_DN_Style)
247     Load.Geometry_XYWH(ButtonRight_X, ButtonRow01_Y, ButtonWidth,
248 ButtonHight)
249     Label.Text('Send NUL')
250 ENDSUB
251
252
253 SUB SendNUL_Up
254     CallSubroutine(SendNUL_Leave)
255     Serial.Tx_NUL(COM2)
256 ENDSUB
257
258
259 ; Button to write STRING -----
260 SUB WriteSTRING_Leave
261     Fill.LabelParameter(PL01_FNL_Button_UP_Style)
262     Load.Geometry_XYWH(ButtonLeft_X, ButtonRow05_Y, ButtonWidth, ButtonHight)
263     Label.Text('Write Output-String')
264 ENDSUB
265
266
267 SUB WriteSTRING_Down
268     Fill.LabelParameter(PL01_FNL_Button_DN_Style)
269     Load.Geometry_XYWH(ButtonLeft_X, ButtonRow05_Y, ButtonWidth, ButtonHight)
270     Label.Text('Write Output-String')
271 ENDSUB
272
273
274 SUB WriteSTRING_Up
275     CallSubroutine(WriteSTRING_Leave)
276     CallSubroutine(InstallKBD1)
277 ENDSUB
278
279
280 ; Button to send STRING -----
281 SUB SendSTRING_Leave
282     Fill.LabelParameter(PL01_FNL_Button_UP_Style)
283     Load.Geometry_XYWH(ButtonMid_X, ButtonRow05_Y, ButtonWidth, ButtonHight)
284     Label.Text('Send Output-String')
285 ENDSUB
286
287
288 SUB SendSTRING_Down
289     Fill.LabelParameter(PL01_FNL_Button_DN_Style)
290     Load.Geometry_XYWH(ButtonMid_X, ButtonRow05_Y, ButtonWidth, ButtonHight)
291     Label.Text('Send Output-String')
292 ENDSUB
```

```

294 SUB SendSTRING_Up
296     CallSubroutine(SendSTRING_Leave)
296     Serial.Tx_String(COM2, OutputString.$)
298 ENDSUB
300
302 ; Button to send first Char of Output-String -----
302 SUB SendFirstChar_Leave
304     Fill.LabelParameter(PL01_FNL_Button_UP_Style)
304     Load.Geometry_XYWH(ButtonRight_X, ButtonRow05_Y, ButtonWidth,
306     ButtonHight)
306     Label.Text('Send First Char')
308 ENDSUB
310
312 SUB SendFirstChar_Down
312     Fill.LabelParameter(PL01_FNL_Button_DN_Style)
312     Load.Geometry_XYWH(ButtonRight_X, ButtonRow05_Y, ButtonWidth,
314     ButtonHight)
314     Label.Text('Send First Char')
316 ENDSUB
318
320 SUB SendFirstChar_Up
320     CallSubroutine(SendFirstChar_Leave)
320     Str.RemoveChar_at_Position(Char.I, OutputString.$, 1)
322     Serial.Tx_Char(COM2, Char.I)
322     CallSubroutine(Draw_OutputField)
324 ENDSUB
326 ;
328 ; -----
330
332 SUB Draw_InputField
332     Load.Geometry_XYWH(InputField_X, InputField_Y, InputFieldWidth.I,
334     InputFieldHight)
334     Fill.LabelParameter(Field_Style)
334     Label.Text(InputString.$)
336
338     FieldHeader_Y.I := InputField_Y - FieldHeaderHight
338     Load.Geometry_XYWH(InputField_X, FieldHeader_Y.I, InputFieldWidth.I,
340     FieldHeaderHight)
340     Fill.LabelParameter(FieldHeader_Style)
340     FieldHeader.$ := 'Input-String'
342     Label.Text(FieldHeader.$)
344 ENDSUB
346
348 SUB Hint_concerning_BackSpace
348     eI.Garbage := InputField_Y + InputFieldHight
350     Load.Geometry_XYWH(InputField_X, eI.Garbage, InputFieldWidth.I, 20)
350     Fill.LabelParameter(Field_Style)
350     Label.Color(transparent)
352     eI.BorderStyle := no_border
352     eI.Position := Pos_right
354     Label.Text('delete last char with backspace')
356 ENDSUB
358
360 SUB Timer4COM
360     Timer.InstallLocal(0, Read_COM)
360     Timer.Load(0, 10)
362     Timer.StartContinuous(0)
364 ENDSUB
364
366 SUB Read_COM
366     LOOP
368     Serial.Rx_Char(COM2, Char.I)

```

```

370     IF Char.I != -1 THEN                ; -1 means empty Buffer
371     ; delete the last Char of InputString when receiving backspace
372     IF Char.I == 0x08 THEN            ; 0x08 means "BackSpace"
373         Str.RemoveLastChar(eI.Garbage, InputString.$)
374     ELSE
375     ; add received char at the end of Input-String
376         Str.AddChar(InputString.$,Char.I)
377         Str.Length(StringLength.I, InputString.$)
378
379     ; make Input-String flow if the Input-String has reached its
380     maximum length
381         IF StringLength.I >= MaxStringLength.I THEN
382             Str.RemoveChar_at_Position(eI.Garbage, InputString.$,
383 1)
384         ENDIF
385     ENDIF
386
387     CallSubroutine(Draw_InputField)
388 ELSE
389     EXITLOOP
390 ENDIF
391 ENDLOOP
392
393 ENDSUB
394
395 ; -----
396 ; OutputField
397 ; -----
398
399 SUB Draw_OutputField
400     Load.Geometry_XYWH(InputField_X, OutputField_Y, InputFieldWidth.I,
401 InputFieldHeight)
402     Fill.LabelParameter(Field_Style)
403     Label.Text(OutputString.$)
404
405     FieldHeader_Y.I := OutputField_Y - FieldHeaderHeight
406     Load.Geometry_XYWH(InputField_X, FieldHeader_Y.I, InputFieldWidth.I,
407 FieldHeaderHeight)
408     Fill.LabelParameter(FieldHeader_Style)
409     FieldHeader.$ := 'Output-String'
410     Label.Text(FieldHeader.$)
411 ENDSUB
412
413 ; -----
414 ; alpha keyboard
415 ; -----
416
417 SUB CallBack_OK
418     OutputString.$ := KBD1.InputLine.$
419     CallSubroutine(Draw_OutputField)
420 ENDSUB
421
422 SUB CallBack_DEL
423     OutputString.$ := ''                ; Delete OutputString
424     CallSubroutine(Draw_OutputField)
425 ENDSUB
426
427
428
429 SUB InstallKBD1
430     ; Keyboard installieren -----
431     Load.Pos_X1Y1(5,40)
432     KBD1.InputLine.$ := 'please enter your text'
433     CallSubroutine(KBD1.DRAW)
434     Load.CallBackAddress ( KBD1.CALLBACK_OK.L,CallBack_OK )      ; CallBack OK
435 installieren
436     Load.CallBackAddress ( KBD1.CALLBACK_CANCEL.L,CallBack_DEL ) ; CallBack
437 DEL installieren
438 ; Load.CallBackAddress ( KBD1.CALLBACK_CHAR.L,CallBack_CHAR ) ; CallBack
439 CHAR installieren
440

```

```

442 ; Load.CallBackAddress ( KBD1.CALLBACK_ENTER.L,CallBack_ENTER ) ;
    CallBack CHAR installieren
444 ; Load.CallBackAddress ( KBD1.CALLBACK_CTRL.L,CallBack_CTRL ) ; CallBack
    CTRL installieren
    ENDSUB
446
448 Styles → ; S T Y L E S
450 SUB Styles
452 Field_Style:
454     INLINENWORDS (no_change) ; entspricht eI.Pos_Xl
455     INLINENWORDS (no_change) ; entspricht eI.Pos_Yl
456     INLINENWORDS (no_change) ; entspricht eI.Width
457     INLINENWORDS (no_change) ; entspricht eI.Height
458     INLINENWORDS (8) ; entspricht eI.SpaceLeft
459     INLINENWORDS (8) ; entspricht eI.SpaceRight
460     INLINENWORDS (0) ; entspricht eI.HorizontalAdjust
461     INLINENWORDS (0) ; entspricht eI.VericalAdjust
462     INLINENWORDS (darkgray) ; entspricht eI.FillColor
463     INLINENWORDS (darkgray) ; entspricht eI.BackColor
464     INLINENWORDS (darkgray) ; entspricht eI.LineColor
465     INLINENWORDS (black) ; entspricht eI.TextColor
466     INLINENWORDS (Pos_left) ; entspricht eI.Position
467     INLINENWORDS (Orientation_0deg) ; entspricht eI.Orientation
468     INLINENWORDS (normal) ; entspricht eI.Appearance
469     INLINENWORDS (color_sunk_1) ; entspricht eI.BorderStyle
470     INLINENWORDS (Font_Arial_12n) ; entspricht eI.FontNumber
471     INLINENWORDS (as_DisplayColor) ; entspricht eI.BackGroundColor
472
474 FieldHeader_Style:
475     INLINENWORDS (no_change) ; entspricht eI.Pos_Xl
476     INLINENWORDS (no_change) ; entspricht eI.Pos_Yl
477     INLINENWORDS (no_change) ; entspricht eI.Width
478     INLINENWORDS (no_change) ; entspricht eI.Height
479     INLINENWORDS (8) ; entspricht eI.SpaceLeft
480     INLINENWORDS (8) ; entspricht eI.SpaceRight
481     INLINENWORDS (0) ; entspricht eI.HorizontalAdjust
482     INLINENWORDS (0) ; entspricht eI.VericalAdjust
483     INLINENWORDS (bisque) ; entspricht eI.FillColor
484     INLINENWORDS (bisque) ; entspricht eI.BackColor
485     INLINENWORDS (bisque) ; entspricht eI.LineColor
486     INLINENWORDS (black) ; entspricht eI.TextColor
487     INLINENWORDS (Pos_left) ; entspricht eI.Position
488     INLINENWORDS (Orientation_0deg) ; entspricht eI.Orientation
489     INLINENWORDS (normal) ; entspricht eI.Appearance
490     INLINENWORDS (color_raised_1) ; entspricht eI.BorderStyle
491     INLINENWORDS (Font_System_9bd) ; entspricht eI.FontNumber
492     INLINENWORDS (as_DisplayColor) ; entspricht eI.BackGroundColor
494 TitelStyle_ExpShadow:
495     INLINENWORDS (no_change) ; entspricht eI.Pos_Xl
496     INLINENWORDS (no_change) ; entspricht eI.Pos_Yl
497     INLINENWORDS (640) ; entspricht eI.Width
498     INLINENWORDS (18) ; entspricht eI.Height
499     INLINENWORDS (8) ; entspricht eI.SpaceLeft
500     INLINENWORDS (8) ; entspricht eI.SpaceRight
501     INLINENWORDS (1) ; entspricht eI.HorizontalAdjust
502     INLINENWORDS (1) ; entspricht eI.VericalAdjust
503     INLINENWORDS (transparent) ; entspricht eI.FillColor
504     INLINENWORDS (as_FillColor) ; entspricht eI.BackColor
505     INLINENWORDS (as_FillColor) ; entspricht eI.LineColor
506     INLINENWORDS (0x5E94) ; entspricht eI.TextColor
507     INLINENWORDS (Pos_left) ; entspricht eI.Position
508     INLINENWORDS (Orientation_0deg) ; entspricht eI.Orientation
509     INLINENWORDS (normal) ; entspricht eI.Appearance
510     INLINENWORDS (no_border) ; entspricht eI.BorderStyle
511     INLINENWORDS (Font_Arial_14n) ; entspricht eI.FontNumber
512     INLINENWORDS (as_Skin_FormBodyColor) ; entspricht eI.BackGroundColor
514 TitelStyle_Exp:
515     INLINENWORDS (no_change) ; entspricht eI.Pos_Xl
516     INLINENWORDS (no_change) ; entspricht eI.Pos_Yl

```

Hauptprogramm

```

518     INLINERWORDS      (640)           ; entspricht eI.Width
519     INLINERWORDS      (18)           ; entspricht eI.Height
520     INLINERWORDS      (8)           ; entspricht eI.SpaceLeft
521     INLINERWORDS      (8)           ; entspricht eI.SpaceRight
522     INLINERWORDS      (0)           ; entspricht eI.HorizontalAdjust
523     INLINERWORDS      (0)           ; entspricht eI.VericalAdjust
524     INLINERWORDS      (transparent)  ; entspricht eI.FillColor
525     INLINERWORDS      (as_FillColor) ; entspricht eI.BackgroundColor
526     INLINERWORDS      (as_FillColor) ; entspricht eI.LineColor
527     INLINERWORDS      (black)       ; entspricht eI.TextColor
528     INLINERWORDS      (Pos_left)    ; entspricht eI.Position
529     INLINERWORDS      (Orientation_0deg) ; entspricht eI.Orientation
530     INLINERWORDS      (normal)     ; entspricht eI.Appearance
531     INLINERWORDS      (no_border)   ; entspricht eI.BorderStyle
532     INLINERWORDS      (Font_Arial_14n) ; entspricht eI.FontNumber
533     INLINERWORDS      (as_Skin_FormBodyColor) ; entspricht eI.BackgroundColor
534
535     ENDSUB
536
537     ; -----
538     ; H A U P T P R O G R A M M -----
539
540     BEGINVIEW
541         EVE.Init()           ; EVE ANNA initialisieren
542
543         Load.Offset_XY(0,0)
544
545         Display.Prepare()
546
547         ; Background -----
548         CallSubroutine ( PL01_Draw_Background )           ; Hintergrund PageLayout 01
549
550         ; Buttons & InputField -----
551         CallSubroutine(PL01_PagelayoutInit)           ; Seite aufbauen mit Titel und
552         Navigation
553
554         CallSubroutine(Layout_Def)
555         CallSubroutine(SetBaudrate_Up)
556         HotSpot.Install(NIL, SetBaudrate_Leave, SetBaudrate_Down, SetBaudrate_Up)
557         CallSubroutine(ClearInput_Leave)
558         HotSpot.Install(NIL, ClearInput_Leave, ClearInput_Down, ClearInput_Up)
559         CallSubroutine(SendCRLF_Leave)
560         HotSpot.Install(NIL, SendCRLF_Leave, SendCRLF_Down, SendCRLF_Up)
561         CallSubroutine(SendNUL_Leave)
562         HotSpot.Install(NIL, SendNUL_Leave, SendNUL_Down, SendNUL_Up)
563         CallSubroutine(WriteSTRING_Leave)
564         HotSpot.Install(NIL, WriteSTRING_Leave, WriteSTRING_Down, WriteSTRING_Up)
565         CallSubroutine(SendSTRING_Leave)
566         HotSpot.Install(NIL, SendSTRING_Leave, SendSTRING_Down, SendSTRING_Up)
567         CallSubroutine(SendFirstChar_Leave)
568         HotSpot.Install(NIL, SendFirstChar_Leave, SendFirstChar_Down,
569         SendFirstChar_Up)
570         CallSubroutine(Draw_InputField)
571         CallSubroutine(Hint_concerning_BackSpace)
572         CallSubroutine(Draw_OutputField)
573
574         CallSubroutine(Timer4COM)
575
576
577         ; HotSpot für ScreenShot -----
578         Load.Geometry_XYWH(600,0,40,40)
579         HotSpot.Install(NIL,NIL,ScreenDump,NIL)
580
581         Display.Show()
582
583         Timer.TableEnable()
584         HotSpot.TableEnable()
585
586         LOOP
587     ENDL00P
588
589     ENDVIEW
590
591

```

7 Anhang

7.1 Anleitung zum Einrichten des Hyper Terminal von Windows XP

Mit Hilfe des „Hyper Terminals“ von Windows XP können Sie mit Ihrem eigerPanel 57 kommunizieren. Dazu muss Ihr PC über die seriellen Schnittstellen mit dem eigerPanel 57 verbunden sein. Wenn Sie hierfür den Ausgang COM1 des eigerPanels 57 verwenden (vgl. Abbildung 1), wird der HyperTerminal auch zum Debug Terminal, auf welchem die Aktivitäten des eigerPanels laufend aufgezeichnet werden.

Das Einrichtungsfenster für den Hyper Terminal öffnen Sie über

Start > Programme > Zubehör > Kommunikation > HyperTerminal .

Geben Sie für die Verbindung einen sinnvollen Namen ein, z.B. „eigerPanel57“
Abbildung 6.



Abbildung 6: Eingangsfenster zur Installation einer neuen Verbindung über die serielle Schnittstelle.

Im nächsten Fenster muss nur die Schnittstelle Ihres PC gewählt werden, über welche Sie die Verbindung herstellen wollen, z.B. COM1 (Abbildung 7).

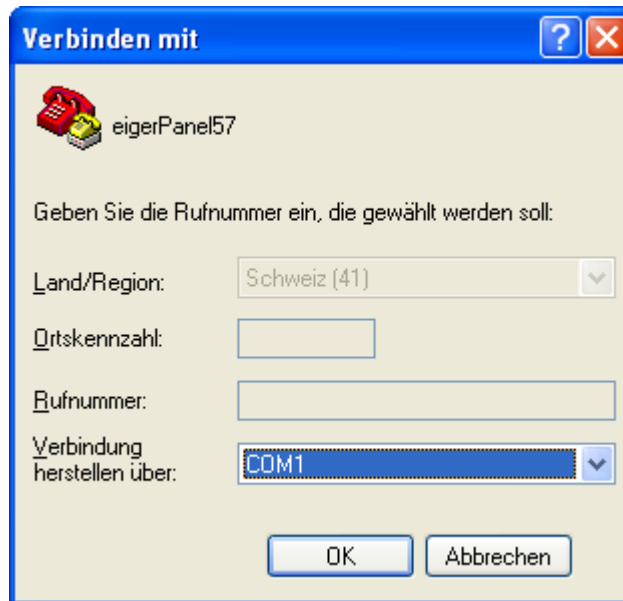
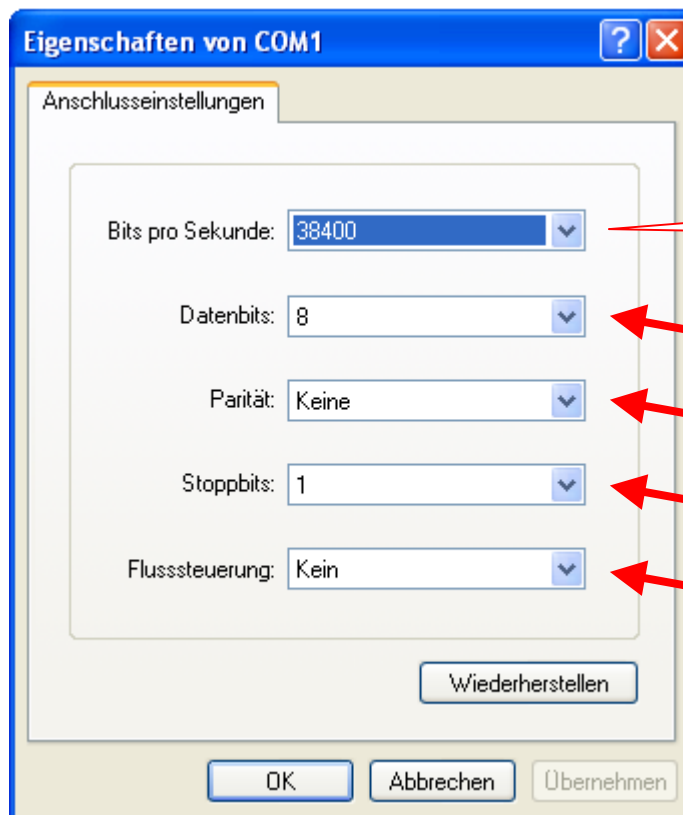



Abbildung 7: Wahl der seriellen Schnittstelle Ihres PC

Für alle weiteren Einstellungen dient das folgende Fenster (Abbildung 8). Die Anzahl Bits pro Sekunde (Baudrate) muss mit der Geschwindigkeit übereinstimmen, welche Sie mit dem Befehl `Serial.SetBaudrate()` in Ihrem eigerScript-Programm festlegen (vgl. Beispiel-Code 1, S.10). Sie können diese Baudrate unter „Eigenschaften“ auch nachträglich verändern, wobei die Verbindung HyperTerminal - eigerPanel für diese Änderung deaktiviert sein muss.



vgl. Tabelle 3, S.9

Abbildung 8: Einstellungen für die neue Verbindung.

Damit ist die Verbindung installiert und bereit für die Kommunikation. Bei aktiver Verbindung (Einstellung in der Symbolleiste: ) wird alles, was Sie im HyperTerminal schreiben direkt ans eigerPanel57 weitergegeben. Wenn Sie diese Einstellungen speichern, können Sie in Zukunft den HyperTerminal für Ihre Verbindung „eigerPanel57“ über

Start > Programme > Zubehör > Kommunikation > HyperTerminal > eigerPanel57.ht
aufrufen.



Sollten Sie im Fenster des HyperTerminals nicht sehen, was Sie schreiben, dann liegt es wahrscheinlich an der aktuellen Konfiguration. Sie können dies regulieren unter Datei > Eigenschaften > Einstellungen > ASCII Konfiguration. Dort können Sie wählen, ob Sie die „einggegebenen Zeichen lokal ausgeben (lokales Echo)“ wollen oder nicht.

7.2 ASCII-Zeichensatz

Der ASCII-Zeichensatz besteht aus 256 Zeichen. Das ist die maximale Anzahl Zeichen, die mit 8 Bit, d.h. einem Byte unterschieden werden können: $2^8 = 256$. Entsprechend ihrer Reihenfolge in der ASCII-Tabelle ist den Zeichen ein ASCII-Wert zwischen 0 bis 255 zugeordnet.

In der IT-Welt ist neben dem Dezimalsystem (10er-Einteilung, d.h. 0-9) auch das Hexadezimalsystem (16er-Einteilung, d.h. 0-F) sehr verbreitet. Je nach Schreibweise kann der ASCII-Wert eines Zeichens unterschiedlich geschrieben werden.

Tabelle 4: ASCII-Werte in unterschiedlichen Schreibweisen

Zeichen	Dezimal	Hexadezimal	eigerScript
NUL	000	00	0x00
0	048	30	0x30
Z	090	5A	0x5A
z	122	7A	0x7A
ß	223	DF	0xDF

Der ASCII-Zeichensatz besteht aus den Sonderzeichen (00-1F), dem Standard-Datensatz (00-1F) und dem erweiterten Zeichensatz (80-FF). Während der Standard-Datensatz fix normiert ist, kann die Zusammenstellung des erweiterten Zeichensatzes variieren. Der ASCII-Zeichensatz, der in eigerScript genutzt wird, ist in Tabelle 5 abgebildet.

Die Steuerzeichen in Tabelle 6 sind applikationsspezifisch und werden je nach Empfängergerät unterschiedlich ausgewertet.

Tabelle 5: ASCII-Codetabelle ohne Steuerzeichen (0x00 bis 0x1F). Die Zeilen- und Spalten-Numerierung entspricht dem Hexadezimalsystem (K hat z.B. den ASCII-Wert 75, in hexadezimaler Schreibweise 4C und in der Schreibweise von eigerScript). ScreenShot des eigerScript-Programms FONT (mitgeliefert als Teil des DemoProgramms TG12, oder als Download unter www.eigergraphics.com).

Tabelle 6: Sonder- und Steuerzeichen des Standard-Datensatzes mit ihren ASCII-Werten

Dez	Hex	Abk.	Beschreibung	Dez	Hex	Abk.	Beschreibung
000	00	NUL	Null-Wert	016	10	DEL	Delete
001	01	SOH	Start Of Header	017	11	DC1	Device Control 1 i (1 ≤ i ≤ 4)
002	02	STX	Start Of Text	018	12	DC2	Device Control 2
003	03	ETX	End Of Text	019	13	DC3	Device Control 3
004	04	EOT	End Of Transmission	020	14	DC4	Device Control 4
005	05	ENQ	Enquiry	021	15	NAK	Negative Acknowledge
006	06	ACK	Acknowledge	022	16	SYN	Synchronous Idle (SYNC)
007	07	BEL	Bell, Signalzeichen	023	17	ETB	End Of Transmission Block
008	08	BS	Backspace, Rückschritt	024	18	CAN	Cancel
009	09	HT	Horizontal Tab, Tabulator in Zeilenrichtung	025	19	EM	End Of Medium
010	0A	LF	Line Feed, Zeilenwechsel	026	1A	SUB	Substitute
011	0B	VT	Vertical Tab, Tabulator in Spaltenrichtung	027	1B	ESC	Escape
012	0C	FF	Form Feed, Blatt wird vom Drucker ausgegeben (Seitenvorschub)	028	1C	FS	File Separator
013	0D	CR	Carriage Return, Wagenrücklauf oder Zeilenumbruch	029	1D	GS	Group Separator
014	0E	SO	Shift Out	030	1E	RS	Record Separator
015	0F	SI	Shift In	031	1F	US	Unit Separator

