LAZ 7. TXD und RXD mit FTDI, Luxus pur

Der Vorteil des Einsatzes der FTDI Komponente ist, dass diese das angeschlossene Gerät automatisch verbinden kann.
Dies kann auf zwei Arten geschehen:
a) DeviceDescriptor
Jeder FTDI Baustein hat eine "Beschreibung" in Textform in seinem Speicher. Diese kann über ein Tool von FTDI auch geändert werden (ftprog, Download bei FTDI)
Wird ein FTDI Chip über USB verbunden, dessen DeviceDescriptor der in der FTDI hinterlegten Kennung entspricht, verbindet sich die Komponente automatisch.
Bei Auslieferung ist diese Kennung "FT232R USB UART"
Entwickelt man eine spezielle Hardware, kann man diese Kennung im FT232 Chip ändern und diese Hardware wir immer sicher erkannt.

b) Die angeschlossene Hardware antwortet auf einen ConnectString
Wird ein neues Gerät an den USB Bus angeschlossen, sendet die FTDI Komponente einen ConnectString (ConnectSendString)
Das angeschlossene Gerät muss dann mit einem ReplyString (ConnectReplyString) antworten, damit es verbunden wird.

Über eine dieser zwei Möglichkeiten kann man zweifelsfrei seine angeschlossene Hardware erkennen und verbinden lassen.
Wir werden jetzt Schritt für Schritt ein solches System entwickeln.

Voraussetzung dafür ist:
ein USB/seriell Kabel mit FTDI Chip oder eine Hardware mit FT232rl Chip on Board.

Arbeitet ihr unter Linux, dann ist der folgende Hinweis sehr wichtig:
Der Linux Kernel bindet ein angeschlossenes FTDI Gerät sofort automatisch in das System ein. Das kann gut sein, ist in unserem Fall aber nicht gut.
Wir haben ja unseren eigenen Treiber (libftd2xx.so.1.3.6 in /user/lib) und möchten selbst Kontrolle über den Chip erlangen. Also müssen wir Linux mitteilen, unser Gerät gefälligst in Ruhe zu lassen.
Dies geht recht einfach, indem wir eine Regel in /etc/udev/rules.d/ mit Namen 98-ftdi.rules erstellen.
Hier der Inhalt dieser Datei. Falls ihr über ftprog den Bezeichner geändert habt, dann muss anstatt "FT232R USB UART" euer neuer Name eingetragen werden! Es ist natürlich auch möglich, weitere Zeilen mit differenten Namen und/oder Vendor/Produkt Kennungen einzufügen.

Quellcode

1. ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", ATTRS{product}=="FT232R USB UART", RUN+="/bin/sh -c 'echo $kernel > /sys/bus/usb/drivers/ftdi_sio/unbind'"

Genug Vorspiel, starten wir ein neues Projekt unter Lazarus (Projekt, Neu, Anwendung, OK)
Platziere ein TFTDI, TShape, TButton, TEDit und TMEmo auf der Form (Form1)

Im Objektinspektor ändern wir:
Shape1.Shape --> stCircle
Edit1.text --> alles rauslöschen
Memo1.lines --> alles rauslöschen
Button1.caption --> "Sende"
Form1.caption --> "COM Test"

Die Positionen und Größen etwas anpassen, dann sollte das so aussehen
Image2.png

Jetzt im Obejktinspektor "Form1" auswählen, dann "Ereignisse" anklicken

Image3.png

Suche nun das Ereignis "onCreate" und klicke doppelt rein. Lazarus erstellt nun die Procedure für Form1.onCreate
Dieses Ereignis tritt auf, wenn unser Fenster "Form1" gestartet/erzeugt wird.
In diesem Ereignis können wir grundsätzliche Dinge regeln beim Start unseres Programms, wie z.B. die Ausgangsfarbe des TShape festlegen.
Shape1 soll uns später signalisieren, ob unser USB Gerät verbunden ist. Beim Start gehen wir erst einmal davon aus, dass nichts verbunden ist.

LAZARUS-Quellcode

```
1. procedure TForm1.FormCreate(Sender: TObject);
2. begin
3. // Füllfarbe des Zeichenobjektes Shape1 auf rot festlegen
4. Shape1.Brush.Color:=clred;
5. // Button1 wird INAKTIV geschaltet. Man kann ihn nicht anklicken!
6. Button1.enabled := false;
7. // wir schalten die FTDI Komponente EIN
8. FTDI1.active := true;
9. end;
```

Die nächsten Schritte sind einfach zu verstehen:
Die Farbe von Shape1 soll sich je nach Verbindungsstatus ändern.
Selektiere im Objektinspektor FTDI1. Schalte auf die Ereignis Seite um. Hier siehst du nun zwei Ereignisse, welche den Connect- und Disconnect Status signalisieren.
Für beide Ereignisse die Proceduren erzeugen und in den Proceduren die Farbe von Shape1, dem Ereignis anpassen.
Außerdem soll man nichts senden können, wenn das Gerät nicht verbunden ist.

LAZARUS-Quellcode

```
1. procedure TForm1.FTDI1Connect(Sender: TObject);
2. begin
3. Shape1.Brush.Color:=clgreen;
4. Button1.enabled := true;
5. end;
6. procedure TForm1.FTDI1Disconnect(Sender: TObject);
7. begin
8. Shape1.Brush.Color:=clred;
9. Button1.enabled := false;
10. end;
```

Alles anzeigen

Nun sollten wir noch die Eigenschaften der FTDI Komponente näher anschauen und einstellen.
Passe zunächst die Baudrate an (19200 Bd)
Wir wollen den DeviceDescriptor unseres FTDI CHip erst einmal nicht verändern und verbinden uns über einen ConnectString.
Die Sendekennung legen wir für "ConnectSendString" fest auf: "Tron alive?" (ohne Anführungsstriche)
Die Empfangskennung für "ConnectReplyString" legen wir fest auf: "Tron alive!" (ohne Anführungsstriche)

Projekt speichern.

Nun benötigen wir ein Programm für unsere Bascom Hardware, welche über das "FTDI Kabel" oder den "FTDI Baustein" verbunden ist:

BASCOM-Quellcode

```
'************************** Sample for bascomforum.de ***************************
'
' Author : Michael Koecher / six1
' created : 2016-12-27
' Version : 1.00
'$regfile = "m8def.dat" 'Chip =17
$regfile = "m8adef.dat" 'Chip =17
$hwstack = 50
$swstack = 50
$framesize = 50
$crystal = 8000000
$baud = 19200
$version 1 , 0 , 0
'Open "Com1:" For Binary As #1
Config Serialin = Buffered , Size = 20 , Bytematch = 13
Dim Rx_str As Byte
Dim Data_in As String * 20
Rx_str = 0
Enable Interrupts
Declare Sub Rx_data()
' *** MAIN ***
Do
' wurde etwas empfangen?
If Rx_str > 0 Then
Rx_str = 0
Call Rx_data
Data_in = ""
End If
Loop
' *** END OF MAIN ***
Sub Rx_data()
If Data_in = "Tron alive?" Then
Print "Tron alive!"
Elseif Data_in = "1" Then
Print "Du hast eine EINS gesendet"
Else
Print "Du hast '" + Data_in + "' gesendet."
End If
Data_in = ""
End Sub
Serial0charmatch:
Input Data_in Noecho
Rx_str = 1
Return
```

Alles anzeigen

Hier kannst du sicher gut erkennen, was beim Empfang von "Tron alive?" passiert...

In diesem Abschnitt hast du einiges Neue gelernt. Selektieren von Komponenten im Objektinspektor, neue Eigenschaften und neue Ereignisse.
Ich hoffe, du hattest bis hierher Spaß an der Sache. Falls etwas nicht so klappen sollte wie das vorgesehen war, dann schreibe einen Kommentar. Ich werde dann versuchen das Problem(chen) zu lösen.
Dadurch können andere auch etwas lernen.