Das Problem

- USB scheint für viele Entwickler sehr kompliziert zu sein
- Oft werden deshalb nur einfache Schnittstellenwandler eingesetzt
- Die Funktionalität von USB kann aufgrund dessen nur in einem sehr kleinen Umfang genutzt werden

Die Lösung

USB Controller mit freien Stacks verwenden!

Vorteile:

- Unabhängigkeit von USB Bausteinen
- Nutzung fertiger Treiber
- Flexibl Reaktionen auf verschiedenste Anforderungen
- Verfügbarkeit vieler USB Funktionalitäten (Übertragungsarten, Standbye-Verhalten, ...)

Um sich schnell und einfach in USB einarbeiten zu können, gibt es auf http://www.usb-projects.net ein Grundkurs zur Vermittlung der wesentlichen Eigenschaften.

USB Stacks

USB Device Stack (usbn2mc):

- Einfache API
- Zahlreiche Beispielprojekte
- Flexible Schnittstelle für Mikrocontroller (ARM7,R8C,AVR)
- USB Baustein USBN9404/03 (National Semiconductor)
- Treiber in der Entwicklung für PDIUSBD12D (NXP) und AT90USB (Atmel)
- http://usbn2mc.berlios.org

USB Host Stack:

- Einfache API
- Eigene Host Controller Treiber (aktuell SL811HS (Cypress))
- Fertige Gerätetreiber (Massenspeichern, HID, HUB)
- Der Debug Monitor unterstützt die Entwicklung
- http://www.embedded-projects.net/usbhost

Projekte

usbprog, programmierbarer Adapter:

- Frimware: AVRISP mkII Klon, ISP, RS232, JTAG,...
- http://www.embedded-projects.net/usbprog

vPort uC Schnittstellen für den PC:

- Linux Treiber / libusb basierte Bibliothek
- Ansteuerbar über das Netzwerk mit http://easyconnect.berlios.de
- http://www.embedded-projects.net/vport

Netzwerkkarte mit TCP/IP Stack:

- Kommunikation mit einem USB Gerät über Sockets
- http://www.ixbat.de/tcp

PS2 zu USB Wandler (HID):

- http://www.ixbat.de/ps2

Logik Analysator unter 15 EUR:

- http://www.ixbat.de/la

AVR USB Bootloader:

- http://www.ixbat.de/boot

USB Debug-Techniken

Linux, usbmon
Es gibt in Linux ein Kernelmodul, das beim Debuggen sehr hilfreich ist. Zu finden ist es unter Device Driver - USB Support - USB Monitor. Mit usbmon kann man sich den Traffic auf dem USB Bus *live* ansehen.
Vorgehensweise:

1. modprobe usbmon
2. mount -t debugfs none_debugs /sys/kernel/debug
3. cat /sys/kernel/debug/usbmon/Xt
   X = Gerätenummer (1,2,3..)

Beispielausgabe: d5ea89a0 3575914555 S Ci:001:00 s a3 00 0000 0003 0004 4 ¡
Ci: (Control IN Transfer über Endpunkt 0 mit Adresse 1), anschliessend folgen die tatsächlichen Daten.

Windows, usbsnoopy
Usbsnoopy funktioniert ähnlich wie usbmon. Jedoch muss man hier zuerst aufzeichnen, um die Daten analysieren zu können, und kann nicht *live* auf den Bus sehen.

`http://www.wingmanteam.com/usbsnoopy/`

libusb - Systemunabhängig

Für den, der auf dem PC betriebssystemunabhängig entwicklen möchte, ist die Bibliothek libusb eine echte Alternative.

`http://libusb.sourceforge.net/`

Mit Hilfe von libusb kann man mit folgenden Betriebssystemen auf den USB Bus zugreifen:

- Linux
- Windows (alle aktuellen Versionen)
- FreeBSD
- NetBSD
- OpenBSD
- Darwin
- MacOS X

Dadurch kann man kompatible Anwendungen und Schnittstellen entwickeln.

Konakt

http://www.usb-projects.net
Benedikt Sauter
sauter@ixbat.de

Schluss mit einfachen USB Schnittstellenwandlern als Ersatz für USB: USB lernen, verstehen und anwenden mit freien USB Stacks.

www.usb-projects.net

FH Augsburg

Benedikt Sauter

`sauter@ixbat.de`

`http://www.usb-projects.net`