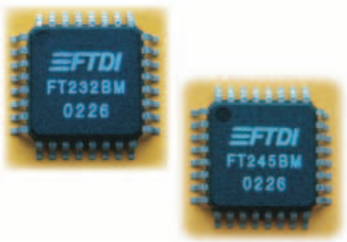
# USB für Mikrocontrollersysteme

*Einfache Implementierung einer USB-Schnittstelle mit den Chips von Future Technology Devices International (FTDI)*

*Florian Skopik*




*Die beiden USB-Chips der Firma*

## Warum USB und nicht COM oder LPT?

Seit Jahren leisten die Serielle Schnittstelle (RS232 bzw. COM) und der Parallel-Port (LPT) gute Dienste bei der Ansteuerung von Mikrocontrollersystemen über den PC. Gründe für einen Wechsel zum *Universal Serial Bus* (USB) gibt es dennoch viele. Hauptsächlich das Aufkommen von „Legacy-Free-PCs" sorgt für ein langsames, aber sicheres Aussterben von COM und LPT. In vielen neuen PC-Systemen (nicht nur Notebooks) werden oft bis zu 6 USB-Schnittstellen standardmäßig angeboten, aber nur noch eine COM; bei hochintegrierten Boards fällt immer öfter selbst diese weg. Da für den Consumerbereich keine neue Hardware für COM oder LPT produziert wird (man denke hier an Drucker, Scanner, Kartenleser, Mäuse etc), verdanken diese beiden Schnittstellen ihr heutiges Dasein nur noch einer kurzen Gnadenfrist, die zwecks Abwärtskompatibilität notwendig ist.

Betrachtet man die aktuelle Marktlage, so scheint USB die Schnittstelle (oder genauer gesagt der Bus) der Zukunft zu sein. Dieser Bus hat sich gegenüber seinen Konkurrenten bis heute auf breiter Front durchgesetzt und jeder Hardwareproduzent hat Produkte mit USB-Interface im Programm (was man z.B. von Firewire (bzw. IEEE1394), Bluetooth etc. nicht behaupten kann).

USB ist im geforderten Rahmen zuverlässig, seit Version 2.0 mit maximal 480MBit/s auch richtig schnell, Plug&Play-fähig und Hot-Pluggable. Aber vor allem ist USB mit seinen einfachen 4-poligen Steckern (der Centronics-Stecker hat 36 Kontakte, der SUB-D der Seriellen Schnittstele 25 oder 9) und daher 4-drahtigen Leitungen so richtig billig.

Die steigende Komplexität der Schnittstelle bedeutet mehr Komfort für den Anwender, fordert aber auch wesentlich mehr Arbeit und Wissen vom Entwickler.

## Die Chips von FTDI im Überblick

Die Firma *Future Technology Devices International* (FTDI) bietet gerade aus diesem Grund spezielle Chips an, den FT232BM und den FT245BM. Diese beiden Produkte sollen dem Entwickler von Mikrocontrollersystemen die Arbeit, sich um das eigentliche USB-Protokoll zu kümmern, abnehmen. Man muss das USB-Protokoll und die Funktionsweise des vergleichsweise komplexen Busses nicht verstehen, um die Schnittstelle zu nutzen.

Die Chips sind eine Art Protokoll-Umsetzer und werden zwischen PC und dem eigentlichen Mikrocontroller geschaltet. Auf $\mu$C-Seite steht ein Standard-UART- oder Parallel-Interface zur Verfügung, auf PC-Seite eine API (*Application Programming Interface*), die sich stark an die Befehle der WinAPI für die COM bzw. Dateibehandlung anlehnt. D.h. man kommuniziert mit den FTDI-Chips so, als würde man eine COM-Schnittstelle ansprechen.

Natürlich kann man auch mehrere Geräte mit FTDI-Chips an einem PC betreiben. Weil aber die USB-Schnittstelle ein Bussystem ist, muss man die einzelnen Geräte an einem Strang voneinander unterscheiden können. Die Identifikation erfolgt durch mehrere Nummern und Strings, die in einem externen EEPROM vom Programmierer abgelegt werden: *Vendor-ID* und *Product-ID* sind zwei 4-stellige Hex-Zahlen, die den Hersteller und den Typ des Gerätes eindeutig festlegen. Weiters wird jedem Gerät eine 8-stellige alphanumerische Seriennummer zugewiesen und ein ID-String vergeben, der beim Einstecken, sowie im Gerätemanager aufscheint. Der restliche Platz im EEPROM kann für eigene Zwecke verwendet werden.

Ein Mikrocontrollersystem mit dem FT2xxBM-Chip und richtig programmierten EEPROM ist vollständig USB1.1/2.0-kompatibel. D.h. es kann mit anderen USB-Geräten am gleichen Bus/Hub betrieben werden, integriert sich vollständig in das Windows-System und taucht im Gerätemanager mit seinem eigenen Namen auf (aus dem EEPROM ausgelesen).

Der FT232BM und FT245BM sind HighTech pur, trotzdem aber auch für Bastler und Schulprojekte sehr gut geeignet. Durch den niedrigen externen Takt (6 MHz; per interner PLL: 48 MHz) sind sie auch aus EMV-Sicht unproblematisch, sind sehr störfest (HF) und funktionieren meist auch bei schlechtem Design/Prototypen (z.B. Single Layer, keine Masseflächen, Fädelleitungen).

Als Nachteil könnte man anführen, dass die Chips trotz ihrer geringen Pin-Anzahl nur in SMD-Bauweise verfügbar sind (LQFP-32 Package). Wer davor zurückschreckt, kann aber gerade für Prototypen auch fertige Module (FT2xxBM, EEPROM, Quarz, USB-Stecker) erwerben, die in einem einfachen DIP-Sockel Platz finden. Darüber hinaus werden auch Evaluation-Boards (mittlerweile auch von anderen Firmen) angeboten, die aus solch einem Modul und einem zusätzlichen Mikrocontroller bestehen (z.B. Microchip PIC).

## Integration in das Mikrocontrollersystem

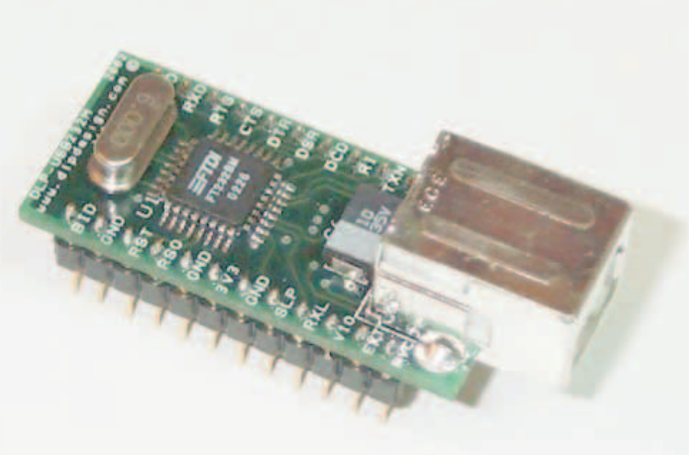
Wie bereits angesprochen gibt es zwei interessante Derivate des USB-Chips der Firma FTDI, den FT232BM (*UART-Interface*) und den FT245BM (*Parallel-Interface*).

Der FT232BM arbeitet mit einer variablen Baudrate von bis zu 1 MBit/s. Er besitzt nicht nur ein Standard UART-Interface (RxD, TxD), sondern simuliert auch das an der RS232 mögliche Hardware-Handshake (RTS/CTS, DTR/DSR, DCD). Er ist damit ideal geeignet, um jedes $\mu$C-System (auch ältere Designs) mit einem UART-Interface auf einfache Weise USB-tauglich zu machen.

In manchen Anwendungsfällen ist die Verwendung des UART-Interfaces allerdings nicht möglich. Sei es, weil man höhere Datenübertragungsraten benötigt, der verwendete $\mu$C kein UART-Interface besitzt, oder dieses für andere Funktionen verwendet wird. Für solche Fälle gibt es den FT245BM. Er stellt auf $\mu$C-Seite ein Standard Parallel-Interface zur Verfügung: einen 8 Bit-Datenbus und 2 Steuerleitungen, die anzeigen, dass neue Daten im Eingangspuffer des Chips verfügbar sind (RXF) oder dass Daten in den Ausgangspuffer geschrieben werden können (TXE). Der FT245BM erreicht am Parallel-Interface eine Geschwindigkeit von bis zu 1 MByte/s.

Der *Universal Serial Bus* bietet im Gegensatz zur COM (RS232) oder LPT auch in Bezug auf das Board-Design einen angenehmen Nebeneffekt: USB-Geräte können über den Bus versorgt werden. Bis maximal 500mA pro Anschluss (bei 5V) an aktiven USB-Hubs (also Hubs mit eigener Versorgung) sind erlaubt. Das

*Fertiges Modul für DIP-Sockel*



http://www.ftdichip.com/

ist für die meisten Anwendungen mehr als genug Power. Man spart so nicht nur Platz auf der Platine, sondern auch ein aufwendiges Schaltnetzteil bzw. ein unhandliches Steckernetzteil. Durch ein geschicktes Design ist es auch möglich, ein Board *Bus-Powered* (über USB) und *Self-Powered* (über externe Versorgung) zu betreiben. Sinnvoll kann dies z.B. bei Geräten sein, die nur ab und zu am PC umprogrammiert oder Daten abgefragt werden, die aber ansonsten im Stand-Alone-Betrieb arbeiten. Man muss nur Sorge tragen, dass keine Rückspeisung von extern in den PC erfolgt. Vor allem bei abgeschalteten PC beträgt die Spannung am USB 0V. Eine Einspeisung könnte den Motherboard-Chipsatz bzw. den USB-Hub beschädigen.

Allgemein gilt bei Experimenten an der USB-Schnittstelle, immer über einen externen aktiven USB-Hub seine Geräte zu testen um im Störfall größere Schäden zu verhindern.

## Integration in die Windows-Umgebung

Auf PC-Seite gestaltet sich der Umgang mit der USB-Schnittstelle genauso einfach wie mit der COM. FTDI stellt zwei grundsätzlich verschiedene Treiber für die gängigsten Betriebssysteme (Win, Linux, MacOS, BSD) zur Verfügung. Die VCP-Treiber (*Virtual COM-Port*) emulieren einen zusätzlichen COM-Port (ähnlich wie manche Modem- oder ISDN-Karten) für jedes angeschlossene Gerät mit FTDI-Chip. Somit kann man sämtliche Software für die RS232, die man schon gewohnt ist (z.B. Windows Hyper-Terminal) auch mit den USB-Chips verwenden. Der VCP-Treiber ist zwar anfangs sehr bequem, vermittelt er doch einen schnellen Einstieg in die Welt des USB ohne viel Aufwand. Er hat aber auch entscheidende Nachteile: Plug & Play ist nur begrenzt möglich, nach dem Einstecken des Gerätes dauert es bis zu 30 Sekunden, bis das USB-Gerät erkannt und in das System integriert wird (also der virtuelle COM-Port eingerichtet wird) und bei mehreren gleichartigen Geräten am selben Bus ist eine eindeutige Identifikation der Geräte nur noch bedingt möglich. Infos aus dem EEPROM können nicht ausgelesen werden und das Gerät taucht im Gerätemanager als zusätzlicher COM-Port und nicht als USB-Device auf. Darüber hinaus ist die maximale Geschwindigkeit bei Verwendung der VCP-Treiber geringer. All diese Probleme umgeht man bei Verwendung nativer USB-Treiber (Direct Driver). Für die Windows-Plattform ist das eine DLL mehr im System-Verzeichnis, dafür genießt man dann alle Vorteile des USB: Hot-Pluggable, Plug & Play, volle Geschwindigkeit.

## Anwendungsentwicklung

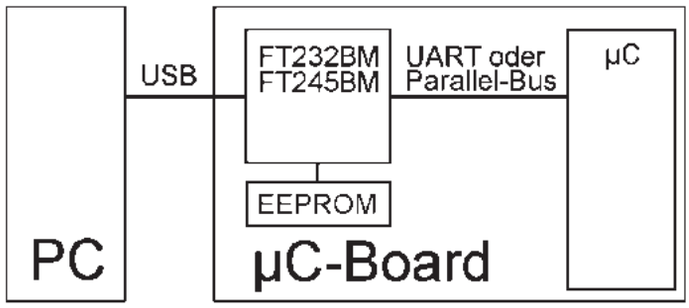
Nun integriert man keine USB-Schnittstelle, nur um sein $\mu$C-System mit einem Terminalprogramm anzusprechen. Meist programmiert man eine spezielle Applikation. Wie man mit den FTDI-Chips bei Verwendung der nativen USB-Treiber kommuniziert, ist in kleinen Demoprogrammen für die meist verwendeten Compiler/IDEs auf der Firmenhomepage zu finden (Visual C++, Borland C++ Builder, Visual Basic, Delphi). Auf der Website findet man auch je ein kleines SDK für diese Compiler. Für den Borland C++ Builder heißt das z.B. eine Bibliothek, ein Header-File und eine HTML-Hilfe, die alle Funktionen erklärt.

Wer sich schon ein wenig mit der Dateiverarbeitung per WinAPI befasst hat, sollte keine Probleme haben, sich sofort zurechtzufinden. Und auch für Neulinge auf dem Gebiet der Windows-Programmierung stellt das Ansprechen der USB-Schnittstelle bei weitem kein unüberwindliches Hindernis dar. Eine Hand voll gut dokumentierter Funktionen reicht aus, um Daten zwischen PC und $\mu$C-System über den FT2xxBM auszutauschen.

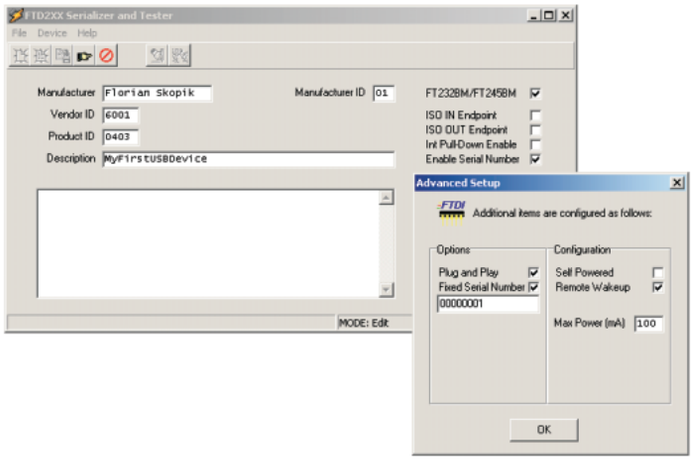
Nach rund 2 Jahren Erfahrung mit den Produkten der Firma FTDI, kann ich aufgrund ihrer Zuverlässigkeit (auch im Langzeitbetrieb über Wochen), Störfestigkeit, der einfachen Programmierung und Beschaltung und wegen des guten Online-Supports (@Fred Dart) jedem Entwickler, Bastler und Interessierten am Thema USB die FT2xxBM-Serie nur weiterempfehlen.

## Weitere Informationen

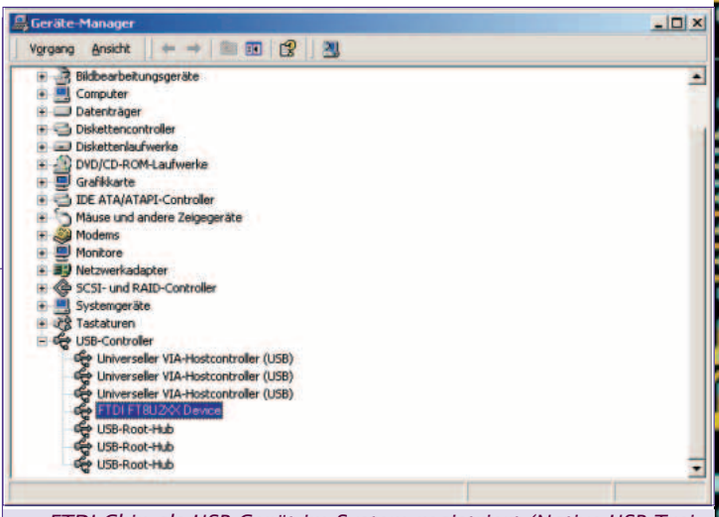
- *http://www.ftdichip.com/*
- *http://www.usb.org/*




*Funktionsweise des FT2xxBM auf Mikrocontrollerboards*




*Programmierung des EEPROMs (In System Programmable über den USB-Chip)*




*FTDI Chip als USB-Gerät im System registriert (Native USB-Treiber)*

## Codefragment in C

```
FT_HANDLE ftHandle;
FT_STATUS ftStatus;
/* Handle auf Device mit Description "MyFirstUSBDevice" holen */
FT_OpenEx("MyFirstUSBDevice",FT_OPEN_BY_DESCRIPTION,&ftHandle);

/* Eigenschaften der Datenübertragung */
FT_SetDataCharacteristics(ftHandle, FT_BITS_8,
FT_STOP_BITS_1,FT_PARITY_NONE);
FT_SetBaudRate (ftHandle, 115200);

/* Lese- und Schreib-Timeouts je 1 Sekunde */
FT_SetTimeouts (ftHandle, 1000, 1000);
/* "Hello USB!" senden */
char message[] = "Hello USB!\0";
ULONG BytesWritten=0;
ftStatus = FT_Write (ftHandle, message, strlen(message),
&BytesWritten);
if((ftStatus != FT_OK) || (BytesWritten != strlen(message))
  cout << "Error writing String!" << endl;
else
  cout << "Everything ok!" << endl;
```