



HOCHSCHULE
MERSEBURG (FH)
University of Applied Sciences

USB

Universal Serial Bus

Michael Hentsche

09453
2INF02

Inhalt:

EINLEITUNG	3
GESCHICHTE	4
VERÖFFENTLICHUNG DER USB-SPEZIFIKATION	5
USB 2.0	6
DER USB UND DIE IEEE-1394.....	6
USB "ON-THE-GO"	6
MINIMALANFORDERUNGEN.....	7
DER HOST-CONTROLLER	7
DAS BETRIEBSSYSTEM	7
DIE KOMPONENTEN.....	8
DIE BUSTOPOLOGIE.....	9
DER USB-PORT	10
DIE AUFGABEN DES HOSTS	10
GRUNDZÜGE DES TRANSFERS	12
TRANSFERTYPEN	13
ENUMERATION: WIE DER HOST GERÄTE ERKENNT	17
ELEMENTE EINES USB-CONTROLLERS	18
WIE KOMMUNIZIERT DER HOST?.....	20
GERÄTETREIBER.....	20
DAS WIN32-TREIBERMODELL (WDM)	20
HUBS: DIE VERBINDUNG ZWISCHEN GERÄTEN UND DEM HOST	21
GRUNDLAGEN:	21
WIEVIELE HUBS LASSEN SICH HINTEREINANDER SCHALTEN?	23
ENERGIEVERWALTUNG	24
SPANNUNGEN	24
STROMBEDARF	24
QUELLENVERZEICHNIS	25

Einleitung

USB (Universal Serial Bus) ist eine schnelle und flexible Schnittstelle, über die Geräte an Computer angeschlossen werden können. Jeder halbwegs moderne PC verfügt üblicherweise über ein paar USB-Anschlüsse. Die Schnittstelle ist vielseitig genug, um sie für Standardperipheriegeräte, wie z.B. Tastaturen und Festplatten, aber auch für spezialisierte Geräte bis hin zu Prototypen oder Unikaten nutzen zu können. USB wurde von Anfang an mit dem Ziel entwickelt, für den Anwender möglichst problemlos einsetzbar zu sein, ohne dass er dafür Hardware oder Software konfigurieren muss.

USB unterscheidet sich stark von den bisherigen Schnittstellen, die er ersetzt. Ein USB-Gerät kann vier verschiedene Transfertypen und drei Geschwindigkeiten nutzen. Beim Anschluss an einen PC muss ein Gerät eine Reihe von Anforderungen beantworten, so dass der PC das Gerät kennen lernt und weiß, wie er mit ihm kommunizieren kann. Im PC müssen alle Geräte über einen Low-Level-Treiber verfügen, der die Kommunikation zwischen Anwendungen und den USB-Treibern des Systems verwaltet.

Geschichte

Nur wenn die Reize der möglichen Verbesserungen gegenüber der Unzweckmäßigkeit und dem Aufwand der nötigen Änderung überwiegt, ist die Einführung einer neuen Schnittstelle gerechtfertigt.

Zuvor war die Entwicklung einer neuen Schnittstelle häufig das Werk eines einzelnen Unternehmens, z.B. Hewlett-Packard: HP-Schnittstellen-Bus (bekannt als GPIB für Laborausrüstung). Doch derartige Schnittstellen stehen unter der ausschließlichen Kontrolle des Entwicklers, was deren Verbreitung erschwert. Daher sind neuere Schnittstellen häufig das Ergebnis einer Zusammenarbeit von Herstellern, die gemeinsames Interesse daran haben. In einigen Fällen fördern Organisationen wie die IEEE oder die TIA Komitees zur Entwicklung von Spezifikationen. Tatsächlich wurden viele der älteren Herstellerstandards von diesen Organisationen übernommen, z.B. war der GPIB Grundlage für den IEEE-488¹.

In anderen Fällen gründen die Entwickler des neuen Standards eine Organisation, die den Standard freigibt und Fragen der Entwicklung behandelt. Dieser Zugang wurde beim USB gewählt.

Das Urheberrecht an der Spezifikation USB 1.1 liegt gemeinsam bei 4 Unternehmen, die alle stark mit Hardware und Software befasst sind: Compaq, Intel, Microsoft, NEC.

Eine frühere Spezifikation mit USB-ähnlichen Merkmalen war der von Phillips und DEC gesponserte ACCESS.bus, bei dem es sich um einen offenen Standard handelte. ACCESS.bus war vom synchronen, seriellen I²C-Bus abgeleitet worden. Obwohl die elektrische Schnittstelle anders aussieht, sind viele der Funktionen und Merkmale USB sehr ähnlich. ACCESS.bus konnte sich im PC-Bereich nie durchsetzen, wird aber bis heute in anderen Bereichen eingesetzt, wie z.B. der intelligenten Batteriesteuerung.

¹ IEEE-488: Auch genannt GPIB (General Purpose Interface Bus). Der Bus ist heute in Laboratorien zur Steuerung und Kontrolle von Meßgeräten noch weit verbreitet.

Veröffentlichung der USB-Spezifikation

Das Release 1.0 im Januar 1996 war das Ergebnis einer mehrjährigen Entwicklung und vorläufiger Releases. Das Release 1.1 im September 1998 behob Probleme vom 1.0-Release und fügte einen neuen Transfertyp hinzu: Interrupt-OUT². Das Release 2.0 erfolgte im April 2000 mit integrierter High-Speed-Option³. Im Dezember 2000 wurden Korrekturen veröffentlicht und ein Neuer Mini-B-Anschluss definiert.



Verschiedene USB-Stecker. Von links nach rechts: Typ A, Typ B, Typ Mini-B

Im PC wurde USB erstmals mit der Freigabe des OEM Service Release 2 von Windows 95 verfügbar. Die USB-Unterstützung wies noch Fehler auf, und es waren nur wenige USB-Peripheriegeräte verfügbar, so dass der USB in dieser Ära nur bedingt Einsatz fand. Verbesserung setzte mit Release von Windows 98 im Juni 1998 ein. Es waren bereits mehr USB-Peripheriegeräte verfügbar und der USB begann als verbreitete Schnittstelle Fuß zu fassen.

Die zweite Ausgabe von Win 98 korrigierte einige Fehler und verbesserte die USB-Unterstützung weiter. Das ursprüngliche Win 98 wurde in Windows 98 Gold umbenannt. Während in Windows NT 4 noch kein USB integriert war, ist dies bei seinen Nachfolgern Windows 2000/XP der Fall, sowie bei Win ME, dem Nachfolger von Win 98.

² Interrupt-OUT: Daten werden vom Host zum Gerät gesendet. Interrupt-IN umgekehrt.

³ Low-Speed-, Full-Speed- und High-Speed-Transfers: USB erlaubt es einem Gerät, mit 1,5 Mbit/s (Low Speed), 12 Mbit/s (Full Speed) oder mit 480 Mbit/s (High Speed, ab USB 2.0 verfügbar) Daten zu übertragen. Diese Raten basieren auf dem Systemtakt der jeweiligen USB-Geschwindigkeit und stellen die physikalische Bitrate dar. Die Datenrate der High Speed Geräte wird momentan durch die Hostcontroller im PC begrenzt und liegt im September 2005 bei etwa 240 MBit/s.

USB 2.0

Ein großer Schritt in der Entwicklung war Version 2.0, die in der Hauptsache sehr viel schnellere Übertragungen verspricht.

Ursprüngliche Erwartungen von 20facher Erhöhung wurden von Untersuchungen und Test übertriften, die zeigten dass eine 40fache Steigerung möglich war.

Übertragungsraten von 480Mbit/s machen USB wesentlich attraktiver für Peripheriegeräte wie Drucker, Scanner, Festplatten und Videoanwendungen.

Der USB und die IEEE-1394

IEEE-1394 oder „FireWire“ ist eine Computerrealisierung von Apple und verfolgt ähnliche Ansätze wie USB. FireWire zwar schneller und flexibler, aber auch teurer. Es eignet sich am besten für Video und andere schnelle Verbindungen. USB eignet sich besonders für Peripheriegeräte mit niedriger bis mittlerer Geschwindigkeit wie z.B. Tastaturen, Drucker, Scanner und externe Laufwerke. Für viele Geräte eignen sich allerdings beide Schnittstellen.

Bei USB steuert ein einzelner Host die Kommunikation mit vielen Peripheriegeräten. Den kompliziertesten Teil der Kommunikation übernimmt der Host, so dass die Elektronik der Peripheriegeräte relativ einfach und preiswert sein kann.

IEEE-1394 dagegen nutzt ein Peer-to-Peer-Modell, bei dem Peripheriegeräte direkt miteinander kommunizieren können oder den Kommunikationsvorgang auch an mehrere Empfänger richten können.

FireWire ist somit eine flexiblere Schnittstelle, bei der aber die Elektronik der Peripheriegeräte komplizierter und teurer ist.

USB "On-the-go"

Durch USB On-the-go (OTG) können entsprechend ausgerüstete Geräte direkt miteinander kommunizieren. Bisher benötigte man bei USB immer einen Computer, der die Host-Funktion übernahm. Durch USB OTG kann jedoch eine direkte Kommunikation zwischen zwei Geräten stattfinden, eines der beiden Endgeräte übernimmt eine eingeschränkte Host-Funktionalität. Mögliche Einsätze sind beispielsweise, Fotos direkt von der Digitalkamera am Drucker auszudrucken oder Musikdateien zwischen zwei MP3-Playern auszutauschen. Kennzeichnet werden USB-OTG-Produkte durch das USB-Logo mit zusätzlichem grünem Pfeil auf der Unterseite und weißem "On-The-Go"-Schriftzug. Die USB-OTG-Spezifikation wurde am 18. Dezember 2001 verabschiedet, inzwischen finden sich immer mehr Produkte, welche USB OTG unterstützen. So kann man inzwischen z.B. bei einigen Festplatten-MP3-Playern auch Fotos von Digitalkameras speichern, ohne dafür einen PC zu benötigen. Dafür muss die Kamera aber dem Standard entsprechen, dass sie auch von modernen Betriebssystemen auch ohne spezielle Treiber als externer Datenträger erkannt werden kann. Etwas unklar ist derzeit noch, ob wirklich alle USB-OTG-Geräte miteinander oder mit anderen USB-Geräten kommunizieren können oder ob das Zusammenspiel sehr beschränkt ist (z. B. nur Produkte des gleichen Herstellers). Bei neuen Geräten ist die Wahrscheinlichkeit dafür sehr groß.

Minimalanforderungen

PCs an die das Peripheriegerät angeschlossen wird, sollen die Schnittstelle auch nutzen können. Dazu benötigen sie Hardware- und Softwareunterstützung. Bei der Hardware handelt es sich um einen USB-Host-Controller und einen Stammhub (Root Hub) mit einem oder mehreren USB-Ports. Für die Software-Unterstützung sorgt ein Betriebssystem mit entsprechenden USB-Funktionen.

Der Host-Controller

Einer Schnittstelle kann kein Erfolg beschieden sein, wenn sie von den PC-Herstellern nicht unterstützt wird. Allerdings haben die Hersteller von PCs und Peripheriegeräten USB geradezu enthusiastisch unterstützt, sodass so gut wie jeder neuere PC einen USB-Controller und mindestens zwei USB-Anschlüsse besitzt. Selbst PCs aus dem Jahre 1997 verfügen meist bereits über USB-Unterstützung. Der „PC 2001 System Design Guide“ von Microsoft und Intel fordert bei einem neuen PC zwei USB-Ports, die für den Anwender zugänglich sind.

Das Betriebssystem

Die Entwicklung eines USB-Peripheriegeräts fällt wesentlich leichter, wenn vorausgesetzt wird, dass der Anwender Windows 98 oder höher einsetzt. Von Windows 95 zu Windows 98 wurde die USB-Unterstützung stark verbessert, und sie können auch nicht die gleichen Gerätetreiber nutzen. Windows NT unterstützt USB überhaupt nicht, es sei denn für das jeweilige Peripheriegerät werden mit Hilfe von Produkten von Drittanbietern spezielle Gerätetreiber entwickelt.

Erwartungsgemäß enthalten DOS und Windows 3.x keinerlei USB-Unterstützung. Linux unterstützt USB 1.x ab den Kernel-Versionen 2.2.18 und 2.4.0 umfassend.

Die Komponenten

Die physischen Komponenten des Universal Serial Bus bestehen aus den Schaltungen, den Steckern und Kabeln zwischen einem Host und einem oder mehreren Geräten.

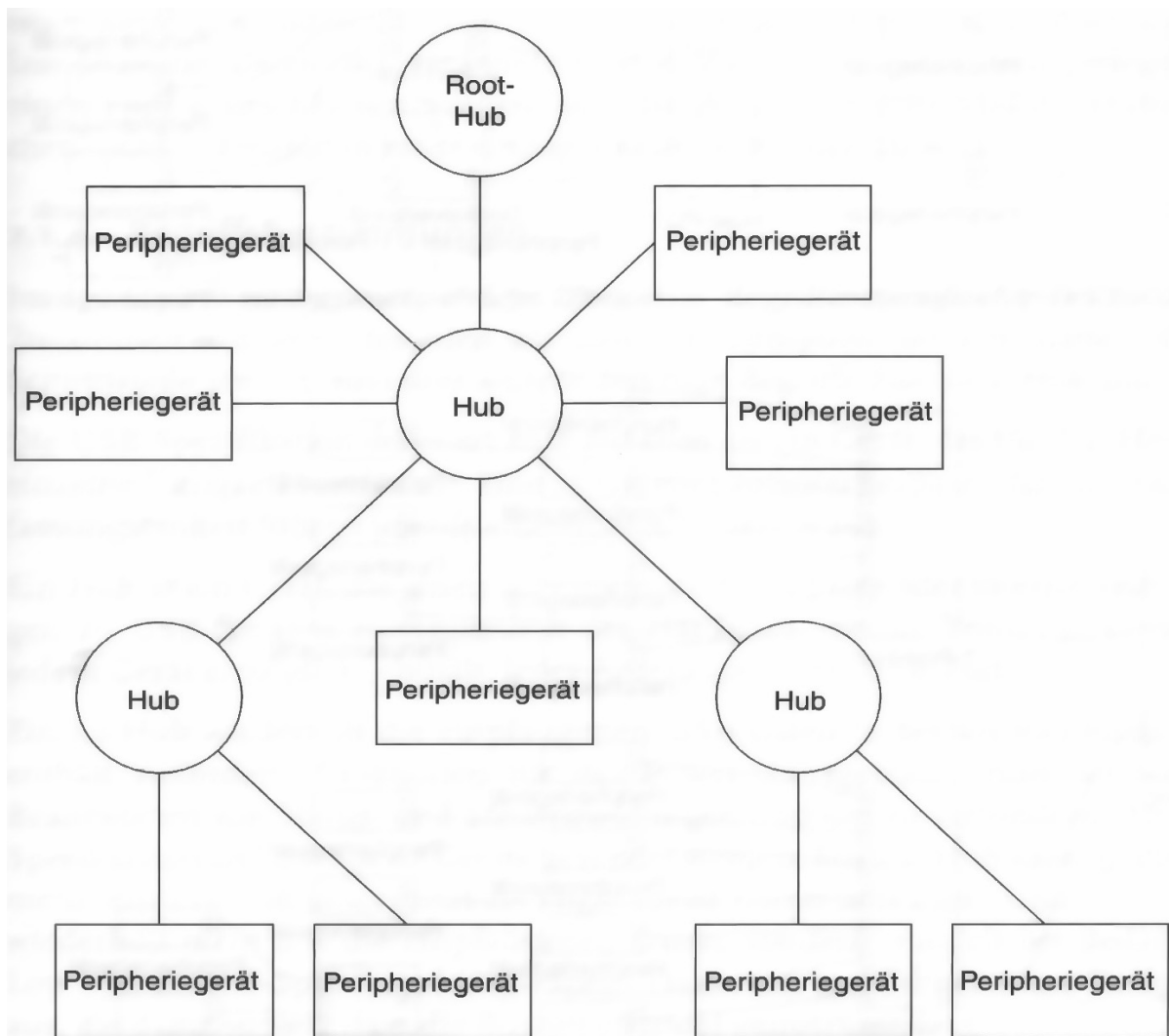
Der Host ist ein Computer, der zwei Komponenten enthält: einen Host-Controller und einen Stammhub (Root Hub). Diese arbeiten zusammen, so dass das Betriebssystem mit den Geräten am Bus kommunizieren kann.

Der Host-Controller formatiert die Daten zur Übertragung auf dem Bus und übersetzt die empfangenen Daten in ein Format, das die Betriebssystemkomponenten verstehen können. Außerdem übernimmt er weitere Funktionen in Bezug auf die Verwaltung der Kommunikation auf dem Bus.

Der Root Hub weist einen oder mehrere Steckverbinder zum Anschluss von Geräten auf, und erkennt zusammen mit dem Host-Controller, wenn Geräte angeschlossen oder entfernt werden, führt Anforderungen des Host-Controllers aus und überträgt Daten zwischen Host-Controller und Geräten. Bei den Geräten kann es sich um Peripheriegeräte als auch um zusätzlich an den Bus angeschlossene Hubs handeln.

Die Bustopologie

Die Anordnung der Verbindungen am Bus ist eine kaskadierte Stern-Topologie (siehe Abbildung). Im Zentrum jedes Sterns findet sich ein Hub. Jeder Punkt an einem Stern ist ein Gerät, das an einen der Ports des Hubs angeschlossen ist. Die Geräte können zusätzliche Hubs oder weitere Peripheriegeräte sein. Die Anzahl der Punkte an jedem Stern kann schwanken, wobei ein typischer Hub zwei, vier oder sieben Ports besitzt. Wenn mehrere Hubs hintereinander geschaltet sind, spricht man von Kaskadierung.



Der kaskadierte Stern beschreibt nur die physischen Verbindungen. Bei der Kommunikation über mehrere Hubs hinweg wissen weder der USB-Gerätetreiber noch das Gerät, ob die Kommunikation über einen oder fünf Hubs abgewickelt wird. Die Hubs verwalten diesen Vorgang automatisch.

Alle Geräte an einem Bus nutzen gemeinsam einen Datenpfad zum Host-Computer. Es kann jeweils immer nur ein Gerät mit dem Host kommunizieren. Falls mehr Bandbreite benötigt wird, kann man den Host durch einen zweiten Datenpfad erweitern, indem man eine Erweiterungskarte mit einem weiteren Host-Controller und Stammhub installiert.

Man kann bis zu fünf Hubs und bis zu insgesamt 127 Peripheriegeräte und Hubs (einschließlich des Stammhubs) hintereinander schalten. Ob es zweckmäßig ist, wenn so viele Geräte einen gemeinsamen Datenpfad nutzen, ist eine andere Frage.

Der USB-Port

USB-Ports unterscheiden sich von vielen anderen Ports dadurch, dass sämtliche Ports am Bus einen gemeinsamen Pfad zum Host nutzen. Bei der seriellen RS-232-Schnittstelle ist jeder Port unabhängig von den anderen. Falls man zwei RS-232-Ports besitzt, hat jeder seinen eigenen Datenpfad, wobei jedes Kabel seine eigenen Daten und nicht die eines anderen Ports überträgt. Die beiden Ports können gleichzeitig senden und empfangen.

Bei USB wurde ein anderer Ansatz gewählt: Jeder Host-Controller unterstützt einen einzelnen Bus bzw. Datenpfad. Jeder Anschluss stellt einen USB-Port dar, wobei sich aber anders als bei RS-232-Ports alle Geräte die verfügbare Zeit teilen. Somit gibt es auch bei mehreren Ports mit jeweils eigenen Steckverbindern und Kabeln nur einen Datenpfad. Es kann jeweils immer nur ein Gerät oder Host senden. Ein einzelner Host kann aber mehrere USB-Host-Controller unterstützen, die jeweils über einen einzelnen Bus verfügen.

Andere Schnittstellen, bei denen ein Datenpfad gemeinsam genutzt wird, sind IEEE-1394 und SCSI.

Die Aufgaben des Hosts

Der Host-PC ist für den Bus verantwortlich. Er muss wissen, welche Geräte an den Bus angeschlossen sind und über welche Fähigkeiten sie jeweils verfügen. Außerdem muss er dafür sorgen, dass alle Geräte am Bus erforderlichenfalls Daten senden und empfangen können. Anwendungsprogramme brauchen sich nicht um die Einzelheiten der USB-Kommunikation zu kümmern. Sie können zum Senden und Empfangen der Daten auf Standard-Betriebssystemfunktionen zurückgreifen, auf die von nahezu jeder Programmiersprache aus zugegriffen werden kann.

- Erkennung von Geräten

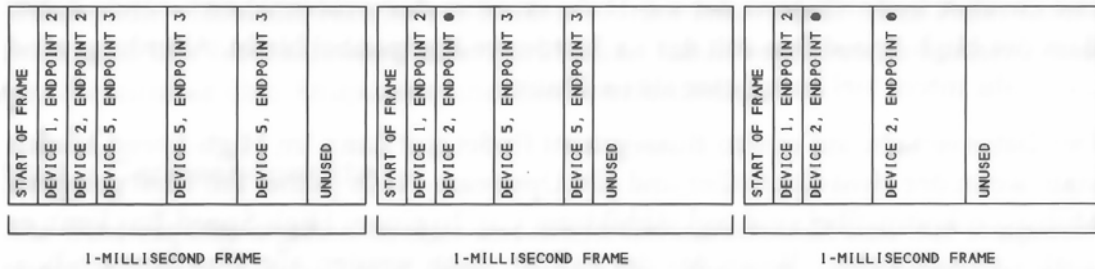
Beim Einschalten melden die Hubs dem Host alle angeschlossenen USB-Geräte. Bei der so genannten Enumeration ordnet der Host allen Geräten Adressen zu und fordert zusätzliche Informationen an. Jedes Mal, wenn nach dem Einschalten ein Gerät entfernt oder angeschlossen wird, erfährt der Host von diesem Ereignis, wobei er neu angeschlossene Geräte enumeriert und abgeklemmte Komponenten aus der Liste der für Anwendungen verfügbaren Geräte entfernt.

- Verwaltung des Datenflusses

Der Host verwaltet den Datenfluss auf dem Bus. Wollen mehrere Peripheriegeräte gleichzeitig Daten senden, unterteilt er die verfügbare Zeit in Segmente, die Frames und Microframes genannt werden. Er unterteilt den Datenfluss in Frames zu einer Millisekunde und stellt jeder Übertragung Teile eines Frames oder Microframes zu Verfügung.

Damit wird sichergestellt, dass Übertragungen, die mit einer spezifischen Rate stattfinden müssen, in jedem Frame die Zeitdauer erhalten, die sie benötigen. Während der Enumeration fordert ein Gerätetreiber die Bandbreite an, die für Übertragungen mit garantiertem Zeitbedarf benötigt wird. Wenn die erforderliche Bandbreite nicht verfügbar ist, lässt der Host keine Kommunikation mit dem Gerät zu und der Treiber kann einen kleineren Teil der Bandbreite anfordern, oder warten bis die angeforderte zur Verfügung steht.

Übertragungen ohne garantierten Zeitbedarf verwenden den verbleibenden Teil der Frames und müssen möglicherweise warten.



- Fehlerüberprüfung

Der Host fügt den übertragenen Daten Fehlerprüfbits hinzu. Wenn ein Gerät oder der Host Daten empfängt und deren Prüfsumme nicht mit den Fehlerprüfbits übereinstimmt, wird der Datenempfang nicht bestätigt, sodass diese Daten erneut gesendet werden.

Der Host kann auch andere Indikatoren empfangen, die darauf hinweisen, dass ein Gerät nicht senden oder empfangen kann, woraufhin er dessen Treiber informieren kann.

- Stromversorgung

Neben seinen zwei Signalleitungen besitzt ein USB-Kabel eine +5V- und Masseleitungen. Der Host arbeitet mit den angeschlossenen Geräten zusammen, um so weit wie möglich Strom zu sparen. Jedes über den Bus mit Strom versorgte Vollleistungsgerät kann pro Bussegment, das mit Strom versorgt wird, mit maximal 500 Milliampere versorgt werden. Ein Bussegment ist die Stromversorgung, die entweder an den Ports eines Host-Controllers oder eines Hubs mit eigener Stromversorgung zur Verfügung gestellt wird.

Ports an einigen batteriebetriebenen PCs und Hubs unterstützen nur Geräte, die maximal 100 Milliampere verbrauchen, Windows unterstützt jedoch keine Hosts mit herabgesetzter Leistungsabgabe.

- Datenaustausch mit Peripheriegeräten

Alle genannten Aufgaben unterstützen die Hauptaufgabe des Hosts, den Datenaustausch mit den Peripheriegeräten. Gerätetreiber könnten eine regelmäßige Kommunikation mit einem Peripheriegerät fordern, während der Host in anderen Fällen nur dann kommuniziert, wenn Anwendungen oder andere Softwarekomponenten entsprechende Anforderungen stellen.

Treten Probleme auf, vermittelt der Treiber diese der entsprechenden Anwendung.

Grundzüge des Transfers

Ein 1.x-Host unterstützt Low-Speed und Full-Speed. Ein 2.0-Host mit Anschlüssen, die dem Anwender zugänglich sind, muss Low-, Full- und High-Speed unterstützen.

Ein 1.x-Hub muss keine Umwandlung der Geschwindigkeiten vornehmen, sondern leitet einfach den Datenverkehr weiter und ändert die Flankenrate der Signale, so dass sie der Geschwindigkeit des Zielgeräts entsprechen.

Ein 2.0-Hub dagegen muss bei Bedarf High-Speed- in Low-Speed- oder Full-Speed-Signale umwandeln können und andere Funktionen ausführen, die der effizienten Nutzung der Buszeit dienen. Die zusätzliche Intelligenz des 2.0-Hubs führt außerdem zur Kompatibilität mit 1.x-Hardware.

Je nachdem, ob sie bei der Anfangskonfiguration oder in Anwendungen verwendet wird, kann man die USB-Kommunikation zwei Kategorien zuordnen:

- Durch die Konfigurationsabfragen lernt der Host das Gerät kennen und bereitet es auf den Datenaustausch vor. Die meisten dieser Abfragen finden statt, wenn der Host das Gerät beim Einschalten oder Anschließen enumeriert.
- Anwendungskommunikation findet dagegen statt, wenn Anwendungen auf dem Host mit enumerierten Geräten Daten austauschen. Dies sind Transfers, über die die eigentlichen Gerätefunktionen ausgeführt werden.

Jeder Transfer besteht aus Transaktionen. Jede Transaktion wiederum besteht aus Paketen mit Informationen.

Alle Übertragungen gehen zu oder von einem sogenannten Geräteendpunkt aus. Der Endpunkt ist ein eindeutig adressierbarer Puffer, in dem einige Bytes Platz finden. Meist handelt es sich dabei um einen Datenspeicher oder ein Register im Controllerchip. In einem Endpunkt gespeicherten Daten können empfangene Daten sein, oder solche die zum Senden bereitstehen. Der Host enthält ebenfalls Puffer für empfangene oder sendebereite Daten, aber keine Endpunkte, stattdessen dient er als Ausgangspunkt für die Kommunikation mit den Geräteendpunkten.

Bevor ein Transfer erfolgen kann, müssen der Host und das Gerät einen Kanal, eine so genannte Pipe einrichten. Eine USB-Pipe ist eine Verknüpfung zwischen einem Geräteendpunkt und der Host-Controller-Software.

Der Host stellt die Kanäle kurz nach dem Einschalten des Systems oder nach dem Anschließen des Geräts her, wenn er von diesem Konfigurationsinformationen anfordert. Falls das Gerät vom Bus entfernt wird, entfernt er auch die nicht mehr benötigten Kanäle. Außerdem kann der Host jederzeit neue Kanäle anfordern oder nicht benötigte entfernen, indem er eine alternative Konfiguration für ein Gerät anfordert. Alle Geräte verfügen über eine Standard-Control-Pipe, die den Endpunkt 0 benutzt und für Setup-Transaktionen verwendet wird.

Transfertypen

Der USB soll laut Konzeption viele Arten von Peripheriegeräten mit unterschiedlichen Anforderungen an Übertragungsrate, Reaktionszeit und Fehlerkorrektur unterstützen. Die vier Datentransfer-Typen entsprechen jeweils unterschiedlichen Anforderungen, wobei ein Peripheriegerät jeweils die Transfertypen unterstützen kann, die sich für seinen Einsatzzweck am besten eignen.

- Control-Transfers:
 - o Funktionen sind durch USB-Spezifikation definiert
 - o Ermöglichen, dass der Host Informationen über ein Gerät liest, die Geräteadresse setzt, und Einstellungen wählt
 - o Müssen von allen USB-Geräten unterstützt werden
- Bulk-Transfers
 - o Wenn Übertragungsrate nicht kritisch ist, die Daten also notfalls warten können
 - o Z.B. Senden einer Datei an einen Drucker, Empfangen von Daten von einem Scanner
 - o Wenn der Bus stark belastet ist durch andere Transfers mit garantierten Übertragungsraten, müssen Bulk-Transfers warten
 - o Wenn sich der Bus im Leerlauf befindet, sind Bulk-Transfers sehr schnell
 - o Können nur von Full-Speed- oder High-Speed-Geräten ausgeführt werden
- Interrupt-Transfers
 - o Bestimmt für Geräte, die periodisch die Aufmerksamkeit des Hosts oder des Geräts benötigen
 - o Können beliebig schnell erfolgen
 - o Abgesehen vom Control-Transfertype die einzige Möglichkeit des Datentransfers für Low-Speed-Geräte
 - o Tastatur und Maus können über Interrupt-Transfer Daten über Tasten- oder Mausbewegungen senden
- Isochrone Transfers
 - o Übertragung innerhalb eines gewissen Zeitraumes wird garantiert
 - o Keine Fehlerkorrektur; unterstützt als einziger Transfertype keine automatische Neuübertragung von fehlerhaft empfangenen Daten
 - o Z.B. Übertragung und Wiedergabe von Audiodaten in Echtzeit
 - o Unterstützt nur durch Full-Speed- oder High-Speed-Geräte

Bulk-, Interrupt- und isochrone Transfers müssen nicht von allen Geräten unterstützt werden, spezifische Geräteklassen können dies aber durchaus fordern.

In der folgenden Tabelle sind Einsatzzweck und Merkmale der Transfertypen zusammengestellt.

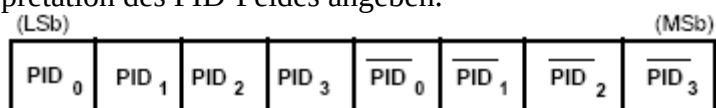
Transfertyp	Control	Bulk	Interrupt	isochron
typische Anwendung	Konfiguration	Drucker, Scanner	Maus, Tastatur	Audio
erforderlich?	ja	nein	nein	nein
bei Low-Speed-Geräten erlaubt?	ja	nein	ja	nein
pro Pipe maximal mögliche Datenbytes/Millisekunde pro Übertragung (High-Speed)	15.872 (31 64-Byte-Transaktionen/ Mikroframe)	53.248 (13 512-Byte-Transaktionen/ Mikroframe)	24.576 (dreier 1024-Byte-Transaktionen/ Mikroframe)	24.576 (dreier 1024-Byte-Transaktionen/ Mikroframe)
pro Pipe maximal mögliche Datenbytes/Millisekunde pro Übertragung (Full-Speed)	832 (13 64-Byte-Transaktionen/ Frame)	1216 (19 64-Byte-Transaktionen/ Frame)	64 (eine 64-Byte-Transaktion/ Frame)	1023 (eine 1023-Byte-Transaktion/ Frame)
pro Pipe maximal mögliche Datenbytes/Millisekunde pro Übertragung (Low-Speed)	24 (dreier 8-Byte-Transaktionen)	unzulässig	0,8 (8 Byte pro 10 Millisekunden)	unzulässig
Richtung des Datenstroms	IN und OUT	IN oder OUT	IN oder OUT (1.0 unterstützt nur IN)	IN oder OUT
reservierte Bandbreite für alle Übertragungen des Typs	10 bei Low/Full-Speed, 20 bei High-Speed (Minimum)	keine	max. 90 bei Low/Full-Speed, max. 80 bei High-Speed (isochron und Interrupt zusammen)	
Fehlerkorrektur?	ja	ja	ja	nein
Nachrichts- oder Datenstrom-Daten?	Nachricht	Datenstrom	Datenstrom	Datenstrom
garantierte Übertragungsrate?	nein	nein	nein	ja
garantierte Latenzzeit (Maximalzeit zwischen Übertragungen)?	nein	nein	ja	ja

Das USB-Protokoll

Paketfeldformate

Alle Pakete beginnen mit einem SYNC-Feld, das von den Eingangs-Schaltkreisen dazu verwendet wird, die eintreffenden Daten dem lokalen Takt anzugleichen. Ein SYNC ist 8 Bits lang bei Full/Low-Speed und 32 Bits lang bei High-Speed-Übertragung. Die letzten zwei Bit in diesem Feld markieren das Ende des Feldes und somit den Anfang der Paket-ID. Es enthält außerdem den „Start-of-Packet“-Begrenzer (SOP). Das SYNC-Feld dient nur der Synchronisation und wird daher nicht in den folgenden Paket-Diagrammen aufgeführt.

Das „Packet Identifier Field“ (PID) folgt stets dem SYNC-Feld eines USB-Pakets. Es besteht aus den vier Bits langen ID, auf die deren vier Bits lange Negation folgt. Dies dient der Fehlerkontrolle an der PID, sodass der Rest des Pakets korrekt interpretiert werden kann. Ein PID-Error entsteht wenn die vier PID-Check-Bits nicht komplement sind mit ihren zugehörigen PID-Bits. PIDs sind aufgeteilt in vier Codegruppen: Token, Data, Handshake und Spezial, wobei die ersten beiden Bits die jeweilige Gruppe, und damit wichtige Eckpunkte zur Interpretation des PID-Feldes angeben.



Das „Address Field“ (ADDR) gibt über ihre Adresse die Funktion an, die entweder die Quelle oder das Ziel eines Datenpaketes ist, abhängig vom Wert der PID des Tokens. Hier können in 7 Bits 128 Adressen formuliert werden. Definitionsgemäß definiert jeder ADDR-Wert eine einzelne Funktion. Bei Reset oder Hochfahren ist der Adresswert standardmäßig null und muss bei der Enumeration durch den Host zugewiesen werden. Der Adresswert null ist als Standardwert reserviert und darf keinem anderen Nutzen zugeordnet werden.

Das „Endpoint Field“ (ENDP) erlaubt flexiblere Adressierung von Funktionen, in welchen mehr als ein Endpunkt verwendet werden soll. Alle Endpunkt-Werte außer null sind funktionspezifisch. Das Endpunkt-Feld ist definiert für IN, SETUP und OUT-Tokens und das PING-Spezialtoken. Alle Funktionen müssen für die Endpunktnummer null eine Control-Pipe unterstützen, die Standard-Control-Pipe. Low-Speed-Geräte unterstützen maximal drei Pipes pro Funktion, nämlich die Standard-Control-Pipe und zwei zusätzliche Pipes (Control/Control, Control/Interrupt, Interrupt/Interrupt). Full- und High-Speed-Funktionen können bis zu 16 IN- und OUT-Endpunkte unterstützen.

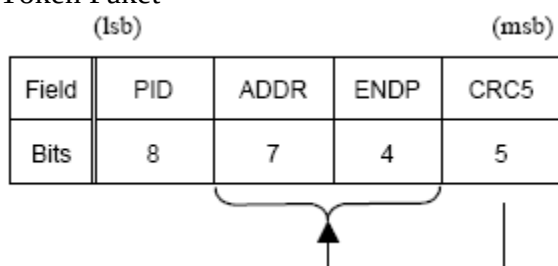
Das „Frame Number“ Feld ist ein 11-Bit-Feld, das pro Frame von Host inkrementiert wird. Der Maximalwert beträgt 7FFH. Das Feld wird nur in „Start of Frame“ (SOF)-Tokens gesendet, bei Beginn eines neuen (Micro)Frames.

Das „Data Field“ kann 0 bis 1024 Bytes groß sein und muss eine ganze Zahl von Bytes enthalten. Die Größe des Datenpakets variiert je nach Transfertyp.

Mit „Cyclic Redundancy Checks“ (CRC) werden alle Paketdaten außer der PID, welche eigene Kontrollbits enthält, kontrolliert. Sie werden beim Senden vor dem Bitstuffing erstellt und beim Empfang nach dem Entfernen der Stopfbits geprüft. Neben dem Token-CRC, kann noch ein Data-CRC vorkommen, der speziell für das Data-Feld herangezogen wird.

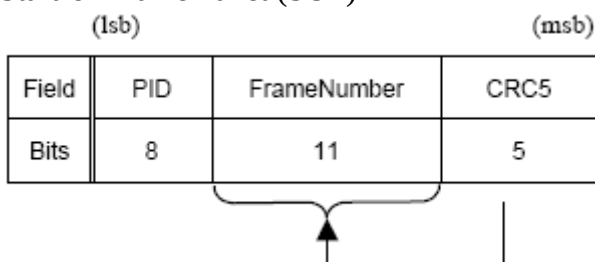
Paketformate

Token Paket



Mit diesem Pakettyp wird festgelegt, welche Transaktion auf dem Bus auszuführen ist.

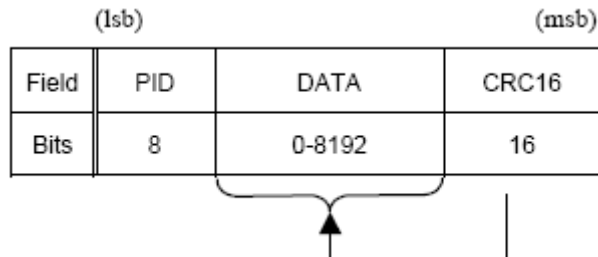
Start-of-Frame Paket (SOF)



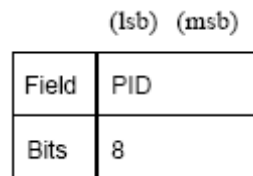
Token- und SOF-Pakete enthalten ein „End-of-Packet“-Feld (EOP) nach je drei Byte Paketdaten. Wenn ein Paket als ein ansonsten gültiges Token oder SOF decodiert wird, aber nicht nach je drei Bytes ein EOP enthält, muss es als ungültig betrachtet und vom Empfänger ignoriert werden.

SOF-Pakete werden durch den Host alle $1\text{ ms} \pm 0,0005\text{ ms}$ auf einem Full-Speed-Bus und alle $125\text{ ms} \pm 0,0625\text{ }\mu\text{s}$ auf einem High-Speed-Bus ausgegeben., um das Erreichen der Grenzen der Frames „auszurufen“, in die der USB-Datenverkehr eingeteilt ist, um parallelen Datenverkehr zu ermöglichen. So können Endpunkte den Framebeginn erkennen und ihre internen Endpunkt-Takte mit dem Host synchronisieren.

Data Paket



Handshake Paket



Das handshake-Paket besteht nur aus der PID, welche hier einen Code enthält, dem ein Handshake-Signal fest zugeordnet ist.

Nur Transaktionen, die Flusskontrolle zulassen, können Handshakes zurückgeben. Sie enthalten ein EOP-Feld nach jedem Byte Paketdaten. Ist dies nicht der Fall, gilt hier dasselbe wie bei Token- und SOF-Paketen. (s.o.)

Zusätzlich sind noch sogenannte „Split Transaction Special Token Packets“ definiert. Split Transaktionen können nur zwischen dem Host und angeschlossenen Hubs ausgeführt werden. Aus Platzgründen möchte ich hier nicht tiefer in diese Thematik eingehen. Ich empfehle, dazu die USB 2.0-Spezifikation zu Rate zu ziehen, die kosten- und registrierungsfrei unter www.usb.org zum Download zur Verfügung steht.

Enumeration: Wie der Host Geräte erkennt

Eine der Aufgaben eines Hubs besteht darin, zu erkennen, wenn Geräte angeschlossen oder entfernt werden. Jeder Hub besitzt einen Interrupt-IN-Kanal⁴, über den er diese Ereignisse dem Host meldet. Beim Booten des Systems fragt der Host seinen Stammbus ab, um zu erfahren, ob irgendwelche Geräte, darunter auch weitere an ihn angeschlossene Hubs mit ihren Geräten, angeschlossen sind. Nach dem Booten fragt der Host weiterhin alle Hubs ab, um über neu angeschlossene oder entfernte Geräte informiert zu werden.

Wenn der Host von einem neuen Gerät erfährt, sendet er eine Reihe von Anforderungen an den Hub des Geräts, und sorgt dafür, dass der Hub einen Kommunikationspfad zwischen dem Host und dem Gerät einrichtet. Dann versucht der Host, das Gerät zu enumerieren, indem er Control-Transfers mit Standard-USB-Anforderungen zum Endpunkt 0 sendet. Dies muss von allen USB-Geräten unterstützt werden. Für eine erfolgreiche Enumeration muss das Gerät alle Anforderungen durch Rückgabe der angeforderten Informationen und weitere angeforderte Aktionen beantworten.

Dies läuft aus Sicht des Anwenders unsichtbar (bis auf eine Erkennungsmeldung) und automatisch ab. Wenn die Enumeration abgeschlossen ist, fügt Windows das neue Gerät zum Gerätemanager in der Systemsteuerung hinzu.

Die Schritte der Enumeration:

1. Der Anwender verbindet ein Gerät mit einem USB-Port.
2. Der Hub erkennt das Gerät.
3. Der Host erfährt vom neuen Gerät.
4. Der Hub stellt fest, ob es sich um ein Low-Speed- oder ein Full-Speed-Gerät handelt.
5. Der Hub setzt das Gerät zurück.
6. Der Host stellt fest, ob ein Full-Speed-Gerät den High-Speed-Modus unterstützt.
7. Der Hub stellt einen Signalpfad zwischen dem Gerät und dem Bus her.
8. Der Host sendet eine Deskriptor-Anforderung, um die maximale Paketgröße des Standardkanals zu ermitteln.
9. Der Host ordnet eine Adresse zu.
10. Der Host ermittelt die Fähigkeiten des Geräts.
11. Der Host ordnet (mit Ausnahme von Kombigeräten) einen Gerätetreiber zu und lädt ihn.
12. Der Gerätetreiber des Hosts wählt eine Konfiguration aus.

⁴ Interrupt-IN: Daten werden vom Gerät zum Host gesendet. Interrupt-OUT umgekehrt.

Elemente eines USB-Controllers

1. Der USB-Port

USB-Controller für Peripheriegeräte müssen natürlich einen USB-Anschluss und unterstützende Schaltkreise besitzen, um mit dem Host kommunizieren zu können. Ein USB-Transceiver sorgt für die Hardware-Schnittstelle zum Bus. Die Schaltkreise, die mit dem Transceiver kommunizieren, bilden eine Einheit mit der allgemeinen Bezeichnung SIE (Serial Interface Engine).

Die SIE behandelt Send- und Empfangsvorgänge von Daten in Transaktionen. Sie interpretiert oder nutzt die Daten nicht, sondern sendet einfach die ihr verfügbaren Daten und speichert alle empfangenen Daten.

Eine typische SIE übernimmt folgende Aufgaben:

- Erkennung ankommender Pakete
- Versenden von Paketen
- Erkennen und Generieren von Start-of-Packet-, End-of-Packet-, Reset- und Resume-Signalen
- Codieren und Decodieren von Daten in das für den Bus erforderliche Format (NRZI mit Bit-Stuffing)
- Prüfen und generieren von Paket-IDs
- Konvertierung serieller USB-Daten in parallele Daten in Registern oder im Speicher und zurück

2. Puffer für USB-Daten

USB-Controller müssen über Send- und Empfangspuffer zur Speicherung gerade empfangener oder zur Übertragung anstehender USB-Daten verfügen. Einige Chips verwenden dafür Register (z.B. der NET2888 von Netchip), während andere (z.B. der EZ-USB von Cypress) Datenspeicherbereiche für Puffer reservieren.

Register, die zu übertragende oder empfangene Daten aufnehmen, werden häufig als FIFO-Puffer strukturiert (First in - First out). Bei jedem Schreibvorgang in ein Send-FIFO wird ein Byte gespeichert, das erst gesendet wird, nachdem alle anderen Bytes, die sich bereits im FIFO-Puffer befunden haben, übertragen worden sind. Ein interner Zeiger auf die nächste zu lesende oder zu schreibende Position wird automatisch inkrementiert, wenn die Firmware in den FIFO schreibt oder aus diesem liest.

Zum Zweck des schnelleren Transfers verfügen manche Chips über doppelte Puffer, die zwei komplette Datensätze in beide Richtungen speichern können. Während ein Datenblock übertragen wird, kann die Firmware so bereits den nächsten Datenblock an eine andere Pufferposition schreiben, so dass die Aktion sofort nach Abschluss der Übertragung des ersten Blocks fortgesetzt werden kann. Die Hardware schaltet automatisch zwischen den beiden Puffern um („Ping-Pong“-Verfahren).

3. CPU

Der Zentralprozessor eines USB-Controllers steuert die Aktionen des Chips durch Ausführung von Befehlen in der im Chip selbst gespeicherten Firmware. Alle CPUs unterstützen einen Befehlssatz, der Anweisungen zum Transport von Daten, der Ausführung mathematischer und logischer Operationen und Sprungbefehle enthält. Der Befehlssatz ermöglicht der CPU auch die Kommunikation mit der SIE (Serial Interface Engine).

4. Programmspeicher

Der Programmspeicher nimmt den von der CPU auszuführenden Programmcode auf. Dieser Speicher kann sich im Prozessorchip selbst oder in einem eigenständigen Chip befinden.

Für den Programmspeicher lassen sich eine Reihe von Speicherarten einsetzen: ROM, EPROM, EEPROM, Flash-EPROM oder RAM. Alle diese Speicherarten, außer nicht batteriegepuffertem RAM, sind nichtflüchtig, sodass die gespeicherten Daten nach dem Abschalten der Stromzufuhr erhalten bleiben. Die Größe des Programmspeichers beträgt mindestens einige Kilobytes. Chips, die auf externen Speicher zugreifen, können ein oder mehrere Megabyte Programmspeicher unterstützen.

Eine andere Bezeichnung für den im Programmspeicher abgelegten Code ist „Firmware“, und sie weist darauf hin, dass dieser Speicher nichtflüchtig ist und sich nicht so leicht ändern lässt wie Programmcode, der ins RAM geladen, editiert, und wieder auf den Datenträger gespeichert werden kann.

5. Datenspeicher

Der Datenspeicher dient als temporärer Speicher während der Programmausführung. Im Datenspeicher lassen die Daten speichern, über USB-Ports übertragen werden sollen oder die anderweitig vom Chip benötigt werden oder von diesem aufbewahrt oder verwaltet werden müssen. Für den Datenspeicher wird üblicherweise RAM in der Größenordnung von 128 bis 1024 Byte verwendet.

6. Register

Register sind eine weitere Option für die temporäre Speicherung. Die meisten CPUs können auf Register schneller als auf anderen Datenspeicher zugreifen.

USB-Controllerchips verfügen üblicherweise über Status- und Kontrollregister, die Informationen über aktivierte Endpunkte, die Anzahl der empfangenen bzw. zu übertragenden Bytes, Informationen zur Fehlerprüfung und andere Informationen über den Einsatz des Chips und den aktuellen Status der übertragenen Daten enthalten.

Die Anzahl der Register und die Einzelheiten ihres Inhalts unterscheiden sich bei den verschiedenen Chipfamilien.

Wie kommuniziert der Host?

Gerätetreiber

Ein Gerätetreiber ist eine Softwarekomponente, die Anwendungen den Zugriff auf Hardwarekomponenten ermöglicht. Die jeweilige Hardware kann sich im Innern des Rechnergehäuses befinden oder über Kabel an den Computer angeschlossen sein. Bei dem Gerät kann es sich um ein Standardperipheriegerät oder auch um spezielle Geräte bis hin zu einzigartigen Prototypen handeln. Einige Gerätetreiber sind Klassentreiber, die eine Kommunikation mit einer Vielzahl von Geräten mit ähnlichen Funktionen ermöglichen.

Gerätetreiber kapseln Daten, so dass sie keine Einzelheiten über die physischen Verbindungen, Signale und Protokolle, die für die Kommunikation mit dem Gerät erforderlich sind, wissen müssen. Durch sie kann der Code von Anwendungen auf Peripheriegeräte einfach über deren Namen (z.B. HP LaserJet) oder über deren Funktion (z.B. ein Joystick) zugreifen. Die Anwendung muss die physische Adresse, an die das Gerät angeschlossen ist, nicht kennen, und sie muss die Handshaking-Signale⁵, die für das Gerät benutzt werden müssen (Busy, Strobe usw.) nicht explizit überwachen und steuern. Anwendungen müssen nicht einmal wissen, ob das Gerät USB oder eine andere Schnittstelle benutzt, da der Anwendungscode für alle Schnittstellen derselbe bleiben kann, wenn die hardwarespezifischen Details auf niedriger Ebene geregelt werden.

Das Win32-Treibermodell (WDM)

USB-Gerätetreiber für Windows müssen dem von Microsoft für den Einsatz unter Windows ab Version 98 definierten Win32-Treibermodell entsprechen. Diese Treiber sind unter der Bezeichnung WDM-Treiber bekannt und haben die Namensendung .sys. Allerdings können auch andere Dateien diese Namensendung besitzen.

Als Low-Level-Treiber verfügen WDM-Treiber über Fähigkeiten, die Anwendungen nicht zur Verfügung stehen, da sie mit dem Betriebssystem auf einer niedrigeren Ebene kommunizieren, auf der ihnen mehr Rechte zustehen, als Anwendungsprogrammen. Ein WDM-Treiber kann Anwendungen den Zugriff auf Geräte erlauben oder verweigern (und so z.B. den Zugriff reservieren) und sind fähig zu DMA-Transfers⁶ und die Reaktion auf Interrupts.

⁵ Handshaking-Signale: Status- und Steuersignale zur Verwaltung des Datenstroms. Beim Hardware-Handshake übertragen eigene Leitungen die Handshake-Informationen, beim Software-Handshake werden diese Code auf den regulären Datenleitungen übertragen.

USB verwendet Hardware-Handshaking.

⁶ DMA-Transfer: DMA-Transfers (Direct Memory Access, direkter Speicherzugriff) laufen ohne zutun der CPU ab, wenn sie erst einmal aufgesetzt sind.

Hubs: die Verbindung zwischen Geräten und dem Host

Alle USB-Geräte müssen an einen Hub angeschlossen werden, ein intelligentes Gerät, das Anschlussstellen für Geräte zur Verfügung stellt und die Verbindungen der einzelnen Geräte mit dem Bus steuert. Geräte, die direkt mit dem PC verbunden werden, werden an den Stammhub angeschlossen.

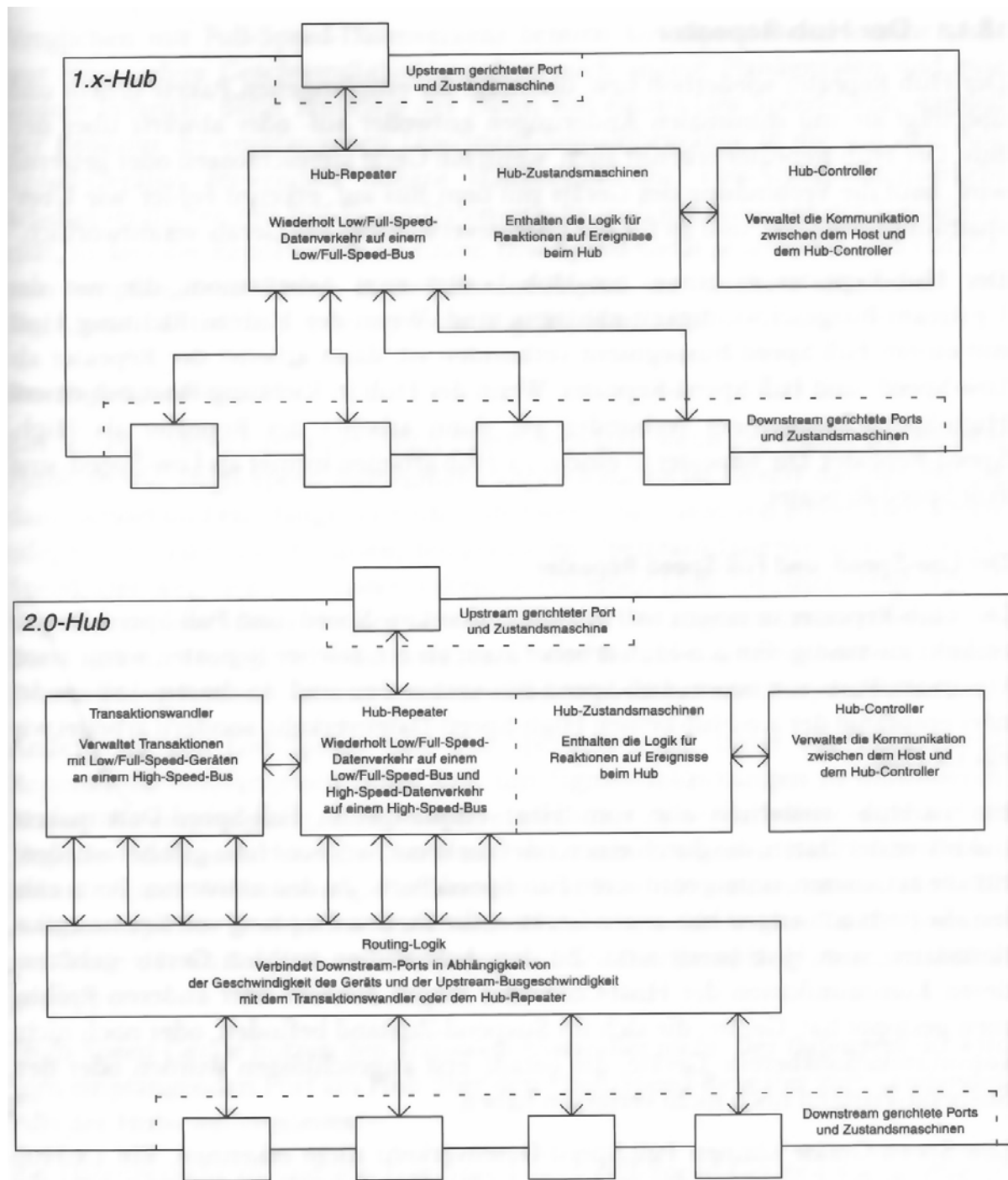
Die zwei Hauptaufgaben eines Hubs sind die Steuerung der Geräteanschlüsse und die Weiterleitung des Datenverkehrs vom und zum Host. Zur Steuerung der Anschlüsse gehört es, sowohl die neu angeschlossenen Geräte in Betrieb zu nehmen und in die Kommunikation einzubinden, als auch Daten von Geräten, die sich falsch verhalten und andere Geräte am Bus stören könnten, zu blockieren. Die Aufgabe des Hubs bei der Weiterleitung der Daten zum und vom Host hängt von der Geschwindigkeit des Busses, des Geräts und der dazwischen liegenden Hubs ab. Ein Hub kann die empfangenen Daten einfach nur wiederholen oder auch den Datenverkehr in eine andere Geschwindigkeit überführen und die Transaktionen mit dem Gerät übernehmen.

Grundlagen:

Jeder externe Hub verfügt über einen Port, über den er mit übergeordneten Geräten (upstream, aufwärts in Richtung Host) verbunden wird. Dieser kann direkt an den Stammhub des Hosts oder einen anderen Hub angeschlossen sein. Alle Hubs verfügen auch über mindestens einen Downstream-Port. Die meisten Downstream-Ports besitzen einen Kabelanschluss. Ausnahmen bilden Hubs, die Teil eines Geräteverbundes (compound devices) sind, deren Ports mit im Gerät eingebetteten Funktionen verbunden sind, z.B. USB-Kopfhörer-Mikrofon-Set oder USB-Kamera-Mikrofon-Kombinationen. Hubs mit zwei, vier und sieben abwärts führenden Ports sind verbreitet. Hubs können über eine eigene Stromversorgung verfügen oder über den Bus vom Host mit Strom versorgt werden, was allerdings gewissen Einschränkungen unterliegt, wie später noch beschrieben wird.

Alle externen Hubs bestehen aus zwei Hauptkomponenten, nämlich einem Hub-Repeater und einem Hub-Controller.

Der Hub-Repeater (Signalumsetzer) ist für die Weiterleitung des USB-Datenverkehrs zwischen dem Stammhub des Hosts oder einem anderen Upstream-Hub und den abwärts angeschlossenen und aktivierten Geräten verantwortlich. Der Hub-Controller steuert die Kommunikation zwischen dem Host und dem Hub-Repeater. Zustandsgesteuerte Geräte erhalten die Schaltlogik, die erforderlich ist, um auf Ereignisse am Hub-Repeater und den Upstream- und Downstream-Ports reagieren zu können. 2.0-Hubs verfügen auch über einen oder mehrere Transaktionswandler und Routing-Logik, die Low-Speed- und Full-Speed-Geräten die Kommunikation über den High-Speed-Bus ermöglicht.



Wieviele Hubs lassen sich hintereinander schalten?

USB ist als Desktop-Bus entwickelt worden und nicht zur Datenfernübertragung. Trotzdem kann man sich fragen, wie weit USB-Peripheriegeräte von ihrem Host entfernt sein können. Die Spezifikation gibt zwar keine maximale Länge für Kabelsegmente an, durch die maximalen Datenlaufzeiten wird die Länge der einzelnen Kabelsegmente aber auf ca. fünf Meter bei Full-Speed und High-Speed und ca. drei Meter bei Low-Speed beschränkt. Man könnte also die Gesamtentfernung zwischen einem Gerät und seinem Hub dadurch erhöhen, dass man eine Reihe von Hubs hintereinander schaltet und diese jeweils mit fünf Meter langen Kabeln verbindet.

Wieviele Hubs lassen sich also in Reihe schalten? Die Anzahl der in Reihe geschalteten Hubs wird durch die elektronischen Eigenschaften der Hubs und der Kabel sowie die Verzögerungen beschränkt, die sich bei der Weiterleitung der Signale über Kabel und Hubs ergeben. Maximal lassen sich fünf Hubs hintereinander schalten, wobei zwischen den einzelnen Hubs und dem letzten Gerät 5-Meter-Kabel verwendet werden können. Das heißt, dass sich ein USB-Peripheriegerät maximal in 30 Meter Entfernung vom Host befinden kann. Wenn es sich um ein Low-Speed-Gerät handelt, liegt das Maximum bei 18 Metern, da das Kabel des Geräts nicht länger als drei Meter sein darf.

Energieverwaltung

Eine der praktischsten Eigenschaften von USB besteht darin, dass Peripheriegeräte über den Bus mit Strom versorgt werden können. Viele Geräte kommen vollständig mit dem vom Bus bereitgestellten Strom aus. Stromversorgung über den Bus bedeutet aber auch, dass die Geräte sich innerhalb der verfügbaren Grenzen bewegen müssen und bei Bedarf in den Energie sparenden Suspend-Modus übergehen können müssen.

Die Fähigkeit, Strom über dasselbe Kabel zu beziehen, das auch die Daten vom und zum PC befördert, hat einige Vorteile. Aus der Sicht des Anwenders sind keine freien Steckdosen in der Nähe des Peripheriegerätes erforderlich und die Geräte werden kleiner und leichter. Aus der Sicht der Hersteller lassen sich solche Geräte kostengünstiger herstellen. Durch die Stromversorgung über den Bus lässt sich oft auch Energie sparen, weil die Netzteile im Rechner effiziente Schaltregler und keine billigen linearen Regler wie in Steckernetzteilen nutzen, über die viele Peripheriegeräte anstelle von integrierten Netzteilen verfügen.

Spannungen

Die nominale Spannung zwischen den Leitungen V(Bus) und GND in einem USB-Kabel beträgt 5V, der tatsächliche Wert kann aber ein wenig höher oder niedriger liegen. Geräte, die Busspannung nutzen, müssen mit diesen Abweichungen zurechtkommen und immer noch den Spezifikationen entsprechen.

Die minimal und maximal zulässigen Spannungen an den von einem Hub abgehenden Schnittstellen betragen:

Hub-Typ	Minimale Spannung	Maximale Spannung
Hohe Spannung	4,75 V	5,25 V
Niedrige Spannung	4,4 V	2,25 V

Vorübergehende Umstände können zudem dafür sorgen, dass die Spannung kurzfristig auf bis zu 4,07 V absinken kann. Wenn bestimmte Komponenten im Gerät höhere Spannungen benötigen, lassen sich im Gerät Schaltregler zur Spannungswandlung einsetzen. Die meisten USB-Controllerchips benötigen eine Spannungsversorgung von +5 oder +3,3 V. Komponenten mit 3,3-V-Spannungsversorgung sind praktisch, weil das Gerät preiswerte, stabilisierte lineare Spannungsregler benutzen kann, um seine Festspannung von 303 V zu erhalten.

Strombedarf

In der Spezifikation werden sowohl Geräte mit niedrigem Strombedarf definiert, die vom Bus mit maximal 100 mA versorgt werden müssen, als auch Geräte mit hohem Strombedarf, die bis zu 500 mA benötigen. Geräte mit eigener Stromversorgung können allerdings so viel Energie benötigen, wie das eigene Netzteil zu liefern imstande ist.

Beim Einschalten können Geräte mit maximal 100 mA über den Bus versorgt werden, bis sie konfiguriert sind. Dadurch können Geräte mit eigener Stromversorgung selbst dann konfiguriert werden, wenn der Anwender das entsprechende Netzteil noch nicht angeschlossen hat. Geräte mit hohem Strombedarf können nicht mehr als 100 mA Strom über den Bus beziehen, bis der Host diesen Bedarf bewilligt hat, was typischerweise während der Enumeration geschieht. Diese müssen also im Modus mit geringem Energiebedarf enumeriert werden können.

Während der Enumeration erfährt der Host die maximale Stromstärke, mit der das Gerät über den Bus versorgt werden muss, und ob es über eine eigene Stromversorgung verfügt oder nicht.

Quellenverzeichnis

Jan Axelson, „USB 2.0, Handbuch für Entwickler“, 2. Auflage 2004, mitp-Verlag Bonn, ISBN 3-8266-1511-5

<http://www.usb.org> - USB-Spezifikation

<http://www.heise.de/newsticker/meldung/9270>

<http://de.wikipedia.org>

<http://lexikon.golem.de>