

USB-Grundlagen

Was wäre, wenn Sie eine von Grund auf neue Peripherieschnittstelle entwerfen könnten? Sicher würde Ihre Wunschliste Folgendes umfassen:

- *Leichte Anwendbarkeit*, so dass man sich keine Gedanken über Einzelheiten der Konfiguration und des Setup zu machen braucht
- *Schnell*, so dass die Schnittstelle nicht zu einem Kommunikationsengpass wird
- *Zuverlässig*, so dass nur selten Fehler auftreten und deshalb eine erneute Übertragung versucht wird
- *Flexibel*, so dass die Schnittstelle für vielerlei Peripheriegeräten genutzt werden kann
- *Preiswert*, so dass Anwender und Hersteller nicht vor dem Preis zurückschrecken
- *Energie sparend*, um Energie zu sparen und das Batterieleben in portablen Computern und Geräten zu verlängern
- *Von Windows und anderen Betriebssystemen unterstützt*, um den Entwicklern das Schreiben hardwarenaher Treiber zur Kommunikation mit den Peripheriegeräten zu ersparen

Der Universal Serial Bus (USB) verfügt über all diese Eigenschaften. Der USB wurde von Grund auf neu als Schnittstelle zur Kommunikation mit vielen Peripheriegerätetypen ohne die Einschränkungen und Ärgernisse älterer Schnittstellen konzipiert.

Jeder halbwegs aktuelle PC und Macintosh-Rechner verfügt über USB-Ports, an die neben Standard-Peripheriegeräten wie z.B. Tastatur, Maus, Scanner, Kamera, Drucker und Laufwerke auch spezielle Hardware für nahezu jeden Einsatzzweck angeschlossen werden können.

In diesem Kapitel soll der USB zunächst mit seinen Vorteilen und Beschränkungen vorgestellt werden, Sie erfahren ein wenig über die Geschichte der Schnittstelle und ihrer jüngeren Erweiterungen und werden sehen, was für den Entwurf und die Programmierung von Geräten mit USB-Schnittstelle benötigt wird.

1.1 Einsatzmöglichkeiten des USB

Der USB kommt am besten überall dort zum Einsatz, wo Computer mit externen Geräten kommunizieren sollen. Die Schnittstelle eignet sich gleichermaßen für Standardperipheriegeräte, die in großen Stückzahlen hergestellt werden, für in kleineren Stückzahlen hergestellte Entwicklungen sowie für einzelne Prototypen.

Damit eine Schnittstelle erfolgreich ist, muss sie den Anforderungen zweier Zielgruppen genügen: denen der Anwender, auf deren Rechnern Anwendungen laufen, die auf die Peripheriegeräte zugreifen, und denen der Entwickler, die die Hardware konstruieren und den Code schreiben, der die Kommunikation mit der Schnittstelle abwickelt. Die Merkmale des USB sind auf die Anforderungen beider Zielgruppen zugeschnitten.

1.1.1 Nutzen für den Anwender

Aus Sicht des Anwenders besteht der Vorteil des USB in der leichten Anwendbarkeit, in schnellen und zuverlässigen Datenübertragungen, in der Flexibilität, in den geringen Kosten und im niedrigen Stromverbrauch. Tabelle 1.1 enthält einen Vergleich des USB mit anderen verbreiteten Schnittstellen.

| Schnittstelle | Format | Max. Anzahl der Geräte | Max. Entfernung in Meter | Max. Geschwindigkeit in Bit/s | Typische Verwendung |
|---------------------------|------------------------------|------------------------|---------------------------------|-------------------------------|---|
| USB | asynchron, seriell | 127 | 5 (oder bis zu 30 m mit 5 Hubs) | 1,5 M, 12 M, 480 M | Maus, Tastatur, Festplatte, Audio, Drucker und andere standardisierte oder spezielle Peripheriegeräte |
| Ethernet | seriell | 1024 | 500 | 10 M/100 M/1 G (10 G) | Allgemeine Netzwerkkommunikation |
| IEEE-1394b (FireWire 800) | seriell | 64 | 90 | 3,2 G | Video, Massenspeicher |
| IEEE-488 (GPIB) | parallel | 15 | 18 | 8 M | Messgeräte |
| IrDA | asynchron, seriell, infrarot | 2 | 1,8 | 16 M | Drucker, Handheld-Rechner |

Tabelle 1.1: Ein Vergleich verbreiteter Computerschnittstellen. Dort, wo ein Standard kein Maximum spezifiziert, sind typische Maximalwerte aufgeführt.

| Schnittstelle | Format | Max. Anzahl der Geräte | Max. Entfernung in Meter | Max. Geschwindigkeit in Bit/s | Typische Verwendung |
|----------------------|------------------------|---|--------------------------|-----------------------------------|--|
| I ² C | synchron, seriell | 40 | 5,5 | 3,4 M | Mikrocontroller-Kommunikation |
| Microwire | synchron, seriell | 8 | 3 | 2 M | Mikrocontroller-Kommunikation |
| MIDI | serielle Stromschleife | 2 (mehr im Flow-Through-Modus) | 15 | 31,5 k | Musik, Steuerung von Bühnenequipment |
| Drucker-Parallelport | parallel | 2 (8 mit Verkettungsunterstützung) | 3 bis 10 | 8 M | Drucker, Scanner, Festplatten |
| RS-232 (EIA/TIA-232) | asynchron, seriell | 2 | 15 bis 30 | 20 k (115 k mit einigen Treibern) | Modem, Maus, Messgeräte |
| RS-485 (TIA/EIA-485) | asynchron, seriell | 32 (vereinzelt 256 mit zusätzlicher Hardware) | 1200 | 10 M | Datenerfassungs- und Steuerungssysteme |
| SPI | synchron, seriell | 8 | 3 | 2,1 M | Mikrocontroller-Kommunikation |

Tabelle 1.1: Ein Vergleich verbreiteter Computerschnittstellen. Dort, wo ein Standard kein Maximum spezifiziert, sind typische Maximalwerte aufgeführt. (Forts.)

Leichte Anwendbarkeit

Die leichte Anwendbarkeit war ein Hauptentwicklungsziel für den USB, und tatsächlich ist das Ergebnis eine Schnittstelle, auf die aus vielen Gründen gern zurückgegriffen wird:

- *Eine Schnittstelle für viele Geräte.* Der USB ist so vielseitig, dass er für viele Arten von Peripheriegeräten genutzt werden kann. Anstatt für jedes Peripheriegerät einen anderen Steckverbinder und andere Protokolle zu verwenden, kann für alle die gleiche Schnittstelle eingesetzt werden.
- *Automatische Konfiguration.* Wenn ein Anwender ein USB-Peripheriegerät anschließt, wird es automatisch vom Betriebssystem erkannt, und die passenden Gerätetreiber werden automatisch installiert. Gegebenenfalls muss beim ersten Anschließen der Komponente ein Datenträger mit den entsprechenden Dateien des Herstellers eingelegt werden. Abgesehen davon erfolgt die Installation aber automatisch. Damit braucht man vor der Verwendung des Peripheriege-

räts weder ein Setup-Programm zu suchen und aufzurufen noch das System neu zu starten.

- *Leicht anzuschließen.* Beim USB braucht das Gehäuse des Computers nicht geöffnet zu werden, um für jedes Peripheriegerät eine Erweiterungskarte hinzuzufügen. Ein typischer PC verfügt bereits über mindestens vier USB-Ports. Die Anzahl der Ports kann durch Anschluss von USB-Hubs (Verteiler) an vorhandenen Ports erweitert werden.
- *Einfache Kabel.* Die Steckverbinder des USB-Kabels haben eine spezifische Form, so dass sie nicht versehentlich verkehrt angeschlossen werden können. Ein Kabelsegment kann bis zu fünf Meter lang sein. Mit Hilfe von Hubs können Verbindungen über maximal 30 Meter hergestellt werden. Die USB-Steckverbinder sind verglichen mit typischen RS-232- und Parallelport-Steckern klein und kompakt. Um einen zuverlässigen Betrieb sicherzustellen, enthält die USB-Spezifikation ausführliche Anforderungen, denen alle Kabel und Steckverbinder genügen müssen.
- *Hot Plugging.* Ein USB-Peripheriegerät kann bei Bedarf beliebig an- oder abgesteckt werden. Dabei spielt es keine Rolle, ob das System und das Peripheriegerät eingeschaltet sind; weder der PC noch das Gerät nehmen dabei Schaden. Das Betriebssystem erkennt, wenn ein Peripheriegerät angeschlossen wird, und bereitet dieses für den Einsatz vor.
- *Keine Anwendereinstellungen.* USB-Peripheriegeräte haben keine durch den Anwender wählbaren Einstellungen wie etwa Port-Adressen und Interrupt-Request-Leitungen (IRQs), so dass keine Jumper (Steckbrücken) gesetzt oder Konfigurationsprogramme ausgeführt werden müssen.
- *Es werden Hardwareressourcen für andere Geräte frei.* Durch die Verwendung des USB für möglichst viele Peripheriegeräte werden IRQ-Leitungen für die Peripheriegeräte frei, die sie wirklich benötigen. Zwar ordnet der PC auch der USB-Schnittstelle eine Reihe von Port-Adressen und eine IRQ-Leitung zu, einzelne Peripheriegeräte benötigen aber keine zusätzlichen Ressourcen und erfordern auch keine Programme, die auf spezifische Port-Adressen zugreifen oder auf die Erkennung von Hardware-Interrupts angewiesen sind. Im Gegensatz dazu erfordern Geräte mit anderen Schnittstellen möglicherweise reservierte Port-Adressen, eine IRQ-Leitung oder auch einen Erweiterungssteckplatz.
- *Keine eigene Stromversorgung erforderlich (teilweise).* Die USB-Schnittstelle enthält Stromversorgungs- und Masseleitungen, die nominal +5 V über das Netzteil des Rechners oder des Hubs zur Verfügung stellen. Peripheriegeräte mit einem Stromverbrauch von weniger als maximal 500 Milliampere können den gesamten Strom über den Bus beziehen und benötigen somit kein eigenes Netzteil. Im Gegensatz dazu hat man bei Peripheriegeräten mit anderen Schnittstellen möglicherweise lediglich die Wahl zwischen einem Netzteil im Gerät oder einem sperrigen und unpraktischen externen Netzteil.

Geschwindigkeit

Der USB unterstützt drei Busgeschwindigkeiten: High-Speed mit 480 MBit/s, Full-Speed mit 12 MBit/s und Low-Speed mit 1,5 MBit/s. Die USB-Hostcontroller in modernen Rechnern unterstützen alle drei Geschwindigkeiten.

Die Busgeschwindigkeiten beschreiben die Rate, mit der die Daten über den Bus übertragen werden. Neben den Nutzdaten müssen jedoch Statussignale, Steuersignale und Fehlerprüfdaten übertragen werden. Zudem müssen alle Peripheriegeräte den Bus gemeinsam nutzen. Daher liegt die mit einem Gerät tatsächlich erreichbare Datentransferrate unterhalb der Busgeschwindigkeit. Die theoretische maximale Datenübertragungsrate für einen einzelnen Transfer liegt bei etwa 53 MByte/s (High-Speed), 1,2 MByte/s (Full-Speed) bzw. 800 Byte/s (Low-Speed).

Low-Speed und Full-Speed wurden in der USB-Spezifikation 1.0 definiert. Low-Speed wurde aus zwei Gründen in die Spezifikation aufgenommen. Mäuse benötigen flexible Kabel, um leicht bewegt werden zu können. Low-Speed-Kabel kommen ohne verdrehte Adern (twisted pair) und starke Abschirmung aus, so dass sie flexibler als Full/High-Speed-Kabel sind. Low-Speed-Geräte lassen sich häufig preiswerter herstellen. Über den Full-Speed-Modus sollten sich die meisten anderen Peripheriegeräte ersetzen lassen, die ansonsten die Serielle (RS-232) oder parallele Schnittstelle benutzt haben. Die bei Full-Speed erreichbaren Datenraten erreichen bzw. übertreffen die mit älteren Schnittstellen erreichbaren Datenraten. Der High-Speed-Modus kam mit der Veröffentlichung der USB-Spezifikation 2.0 als zusätzliche Option hinzu.

Zuverlässigkeit

Die Zuverlässigkeit des USB beruht sowohl auf der Hardware als auch auf den für den Datentransfer verwendeten Protokollen. Die Hardwarespezifikationen für USB-Treiber, -Empfänger und -Kabel eliminieren das meiste Rauschen, das andernfalls Datenfehler verursachen könnte. Das USB-Protokoll ermöglicht die Erkennung von Fehlern in den empfangenen Daten und kann den Sender benachrichtigen, so dass die Daten erneut übertragen werden können. Das Erkennen, Benachrichtigen und erneute Übertragen erfolgt hardwaregestützt und erfordert keine Programmierung oder Anwendereingriffe.

Niedrige Kosten

Auch wenn der USB komplizierter als frühere Schnittstellen ist, sind die Komponenten und Kabel preiswert. Ein Gerät mit einer USB-Schnittstelle kostet meist genauso viel oder sogar weniger als ein Pendant mit älterer Schnittstelle oder einer aktuelleren Schnittstelle, wie z.B. IEEE-1394.

Niedriger Stromverbrauch

Stromsparschaltungen und -code können nicht benutzte USB-Peripheriegeräte automatisch abschalten. Dennoch können sie bei Bedarf weiterhin reagieren. Der verringerte Energieverbrauch spart Kosten, schont die Umwelt und ermöglicht bei batteriebetriebenen Geräten längere Laufzeiten zwischen den Ladevorgängen.

Drahtlose Kommunikation

USB war ursprünglich eine Schnittstelle für verkabelte Geräte. Mittlerweile existieren aber auch Optionen für kabellose Geräte, die USB bei der Kommunikation mit dem PC nutzen.

1.1.2 Vorteile für Entwickler

Viele der oben beschriebenen Vorteile für den Anwender erleichtern auch die Arbeit des Entwicklers. Zum Beispiel bedeuten die für den USB definierten Kabelstandards und die automatische Fehlerprüfung, dass sich die Entwickler keine Gedanken über das Spezifizieren von Kabeleigenschaften oder über die Bereitstellung einer softwareseitigen Fehlerprüfung zu machen brauchen.

USB besitzt weitere Vorzüge, von denen Entwickler profitieren. Zu den Entwicklern zählen z.B. Hardwarekonstrukteure, die die Bauteile für Geräte auswählen und daraus Schaltungen entwickeln, PC-Programmierer, die die im Gerät eingebettete Software schreiben, und Programmierer, die die Programme zur Kommunikation mit den USB-Geräten schreiben.

Die Entwickler profitieren von der Flexibilität des USB-Protokolls, der Unterstützung hinsichtlich der Controllerchips und des Betriebssystems und der im *USB Implementers Forum* (USB-IF) gebotenen Unterstützung.

Vielseitigkeit

Durch die vier Transfertypen und drei Geschwindigkeiten eignet sich der USB als Schnittstelle für viele Peripheriegerätetypen. Es gibt Transfertypen, die sich für den Austausch großer und kleiner Datenblöcke mit und ohne Zeitbeschränkungen eignen. Für Daten, die keine Verzögerung dulden, kann der USB Bandbreiten oder Obergrenzen für Zeitspannen zwischen Übertragungen garantieren. Diese Fähigkeiten werden insbesondere unter Windows begrüßt, da hier der Echtzeit-Zugriff auf Peripheriegeräte oft eine Herausforderung darstellt. Auch wenn das Betriebssystem, Gerätetreiber und Anwendungsprogramme immer noch unvermeidbare Verzögerungen hervorrufen können, sorgt USB dafür, dass Transfers nahezu in Echtzeit erfolgen können.

Anders als andere Schnittstellen ordnet der USB den Signalen keine spezifischen Funktionen zu und setzt auch nichts anderes über die Verwendung der Schnittstelle voraus. Demgegenüber wurden die Status- und Steuerleitungen am PC-Par-

allelport beispielsweise im Hinblick auf die Kommunikation mit Zeilendruckern definiert. Die Schnittstelle besitzt fünf Eingangsleitungen, denen jeweils eine Funktion wie etwa die Angabe eines Belegt- oder eines Papierausgabestatus zugeordnet ist. Als die Entwickler begannen, den Port für Scanner und weitere Peripheriegeräte zu nutzen, die große Datenmengen an den PC senden, stellte die Beschränkung auf nur fünf Eingänge ein Hindernis dar. (Schließlich wurde die Schnittstelle erweitert, so dass sie 8-Bit-Eingaben zulässt.) USB trifft keine derartigen Annahmen und eignet sich für nahezu alle Arten von Peripheriegeräten.

Für die Kommunikation mit den üblichen Gerätetypen, wie z.B. Drucker, Tastatur und Laufwerke, existieren beim USB definierte Klassen mit Angaben über Geräteanforderungen und Protokolle. Entwickler können die Klassen als Richtlinie verwenden, ohne alle Einzelheiten jeweils neu entwickeln zu müssen.

Betriebssystemunterstützung

Windows 98 war das erste Windows-Betriebssystem mit zuverlässiger USB-Unterstützung und auch die Nachfolger Windows 2000, Windows Me, Windows XP und Windows Server 2003 unterstützen ihn. Auch wenn sich dieses Buch auf die Windows-Programmierung für PCs konzentriert, verfügen auch andere Computer und Betriebssysteme über USB-Unterstützung, wie z.B. Apple Macintosh-Rechner und das Betriebssystem Linux für PCs. Und auch einige Echtzeit-Kernel unterstützen USB.

Die Behauptung, dass ein Betriebssystem USB unterstützt, kann vielerlei bedeuten. Auf unterster Ebene muss ein Betriebssystem mit USB-Unterstützung drei Aufgaben bewältigen:

- erkennen, wenn Geräte an das System angeschlossen oder von ihm getrennt werden
- mit neu angeschlossenen Geräten kommunizieren, um zu ermitteln, wie die Daten ausgetauscht werden können
- einen Mechanismus bereitstellen, über den die Softwaretreiber mit der USB-Hardware des Rechners und mit den Anwendungen, die auf die USB-Peripheriegeräte zugreifen, kommunizieren können

Auf einer höheren Ebene kann Unterstützung durch ein Betriebssystem auch bedeuten, dass Klassentreiber integriert wurden, über die Anwendungsprogrammierer auf Geräte zugreifen können. Wenn das Betriebssystem keine geeigneten Treiber für ein spezifisches Peripheriegerät enthält, müssen sie von den Herstellern der Geräte bereitgestellt werden.

Mit jeder neuen Windows-Version hat Microsoft neue Klassentreiber hinzugefügt. Zu den mittlerweile unterstützten Gerätetypen zählen Human Interface Devices (HIDs: Tastatur, Maus, Spielesteuerungen usw.), Audio-Komponenten, Modems,

Kameras (Standbildaufzeichnung) und Scanner, Drucker, Massenspeichergeräte (Disketten, Festplatten, CD/DVD-Laufwerke usw.) und Smart-Card-Lesegeräte. Filtertreiber können gerätespezifische Funktionen und Merkmale innerhalb einer Klasse unterstützen. Anwendungen können API-Aufrufe (Application Programming Interface) oder andere Komponenten des Betriebssystems zur Kommunikation mit den Gerätetreibern nutzen.

Für Geräte, die in keine der unterstützten Klassen fallen, bieten einige Hersteller von USB-Controllern für Peripheriegeräte Treiber an, die Entwickler in Verbindung mit den Controllern des Herstellers verwenden können.

USB-Gerätetreiber nutzen das WDM (Windows Driver Model – Windows-Treibermodell), das eine Architektur für Treiber definiert, die unter Windows 98 und seinen Nachfolgeversionen funktionieren. Dadurch sollen Entwickler mehrere Windows-Versionen mit einem einzigen Treiber unterstützen können. Dieses Ziel wird praktisch jedoch nicht ganz erreicht. Für viele Geräte werden weiterhin mehrere verschiedene WDM-Treiber benötigt, wie z.B. einer für Windows 98/Me und einer für Windows 2000/XP. Da Windows bereits Low-Level-Treiber enthält, die die Kommunikation mit der USB-Hardware übernehmen, lassen sich USB-Treiber üblicherweise leichter schreiben als Treiber für Geräte mit anderen Schnittstellen.

Unterstützung von Peripheriegeräten

Auf Seiten der Peripheriegeräte muss die Hardware jedes USB-Geräts einen Controllerchip enthalten, der die Details der USB-Kommunikation abwickelt. Einige Controller sind vollständige Microcontroller, die CPU, Programm- und Datenspeicher und eine USB-Schnittstelle enthalten. Andere Controller müssen mit einer externen CPU verbunden werden, die bei Bedarf mit dem USB-Controller kommuniziert.

Das Peripheriegerät ist für das Beantworten von Anforderungen zum Senden und Empfangen von Daten zur Erkennung und Konfiguration des Geräts und für das Empfangen und Senden weiterer Daten über den Bus verantwortlich. In einigen Controllern sind einige Funktionen hardwaremäßig mikrocodiert, so dass sie nicht programmiert zu werden brauchen.

Viele USB-Controller basieren auf verbreiteten Architekturen wie etwa auf Intels 8051 oder PICMicro• von Microchip Technology, enthalten aber zusätzliche Schaltungen und Maschinencode zur Unterstützung der USB-Kommunikation. Wenn Sie bereits mit einer Chip-Architektur vertraut sind, von der es eine USB-fähige Variante gibt, dann brauchen Sie keine völlig neue Chip-Architektur zu erlernen. Die meisten Hersteller von Peripheriegeräten liefern Beispielcode für ihre Chips. Durch Verwendung dieses Codes als Basis für eigene Entwicklungen lässt sich viel Zeit sparen.

USB-Entwicklerforum

Bei einigen Schnittstellen bleiben Sie bei der Entwicklung und Inbetriebnahme weitgehend auf sich allein gestellt. Bei USB werden eine Menge zusätzlicher Hilfestellungen im *USB Implementers Forum* (USB-IF) bzw. dessen Website (www.usb.org) angeboten. Das USB-IF ist eine nicht-kommerzielle Organisation, die von den Unternehmen gegründet wurde, die die USB-Spezifikation entwickelt haben.

Das Ziel des Forums ist die Unterstützung der Fortschritte und der Akzeptanz der USB-Technologie. Zu diesem Zweck bietet das USB-IF Informationen, Hilfsmittel und Tests an. Zu den verfügbaren Informationen zählen die Dokumente mit den Spezifikationen, White Papers, FAQs und ein Diskussionsforum, in dem Entwickler öffentlich Fragen zum Thema USB diskutieren können. Zu den vom USB-IF zur Verfügung gestellten Hilfsmitteln zählen Software und Hardware, die das Entwickeln und Testen von USB-Komponenten unterstützen. Die Unterstützung von Tests umfasst die Entwicklung von Kompatibilitäts- und Funktionstests und Workshops, in denen Entwickler ihre Produkte testen lassen können. Wenn die Produkte die Tests bestanden haben, dann dürfen sie das USB-Logo tragen.

1.1.3 Jenseits des Rummels

Die vielen Vorzüge des USB machen ihn zu einem aussichtsreichen Kandidaten für viele Peripheriegeräte. Allerdings eignet sich eine Schnittstelle auch nicht für alle Aufgabenstellungen.

Beschränkungen der Schnittstelle

Alle Schnittstellen weisen Beschränkungen auf, die ihren Einsatz in einigen bestimmten Anwendungsbereichen unpraktisch werden lassen. Bei USB gilt es, Beschränkungen hinsichtlich Geschwindigkeit und Entfernung, die mangelnde Unterstützung von Peer-to-Peer-Kommunikation, fehlende Möglichkeiten zum gleichzeitigen Versenden von Daten an mehrere Empfänger und die mangelnde Unterstützung älterer Hardware und Betriebssysteme zu beachten.

Geschwindigkeit. USB ist zwar vielseitig, eignet sich aber auch nicht für alle Anwendungen. Der High-Speed-Modus kann zwar mit der IEEE-1394-Schnittstelle (FireWire; 400 MBit/s) konkurrieren, ist IEEE-1394b mit seinen noch höheren Übertragungsraten von 3,2 GBit/s aber unterlegen.

Entfernung. Der USB wurde als Desktop-Bus entwickelt, wobei davon ausgegangen wurde, dass die Peripheriegeräte nicht weit entfernt sind. Ein Kabelstück kann maximal etwa fünf Meter lang sein. Andere Schnittstellen wie etwa RS-232, RS-485, IEEE-1394b und Ethernet erlauben wesentlich längere Kabel. Immerhin kann die mit einer USB-Verbindung überbrückbare Entfernung mit fünf Hubs und entsprechendem Kabel auf 30 Meter vergrößert werden.

Eine Möglichkeit zur Überwindung der 30-Meter-Grenze besteht darin, am PC zwar eine USB-Schnittstelle zu verwenden, das Signal für die Fernverkabelung und für die Peripherieschnittstelle aber für eine RS-485- oder andere Schnittstelle umzuwandeln.

Peer-to-Peer-Kommunikation. Jede USB-Kommunikation findet zwischen einem Host-Computer und einem Peripheriegerät statt. Beim Host handelt es sich um einen PC oder einen anderen Computer mit Hostcontroller-Hardware. Das Peripheriegerät enthält die Geräte-Controller-Hardware. Weder Hosts noch Peripheriegeräte können direkt miteinander kommunizieren. Bei anderen Schnittstellen, wie z.B. bei IEEE-1394, können Peripheriegeräte auch direkt miteinander kommunizieren.

Eine Teillösung dieses Problems bietet USB On-The-Go. USB-On-The-Go definiert einen Host-Computer mit verringerten Fähigkeiten, der in eingebetteten Geräten eingesetzt werden kann und an den nur ein einziges USB-Peripheriegerät angeschlossen werden muss.

Broadcasting. USB bietet keine Möglichkeit für das gleichzeitige Versenden einer Mitteilung an mehrere an den Bus angeschlossene Geräte. Der Host muss den Geräten die Mitteilung individuell zusenden. Wenn Broadcasting möglich sein muss, dann können Sie auf IEEE-1394 oder Ethernet zurückgreifen.

Legacy-Hardware. Ältere (»Legacy«-) Computer und Peripheriegeräte besitzen keine USB-Ports. Wenn Sie derartige Peripheriegeräte an einen USB-Port anschließen wollen, dann stellen Wandler (Konverter/Adapter) zur Übersetzung zwischen USB und der älteren Schnittstelle eine Lösung dar. Verschiedene Hersteller bieten Wandler für Peripheriegeräte mit RS-232-, RS-485- und Centronics-Schnittstelle (parallele Druckerschnittstelle) an. Allerdings sind derartige Konverter nur für Peripheriegeräte sinnvoll, die über konventionelle Protokolle kommunizieren und vom Gerätetreiber des Konverters unterstützt werden. Die meisten Konverter für die parallele Schnittstelle unterstützen lediglich die Kommunikation mit Druckern. Konverter, die die meisten Geräte mit RS-232-Schnittstelle unterstützen, sind allerdings erhältlich.

Wenn Sie ein USB-Peripheriegerät mit einem PC verbinden wollen, der USB nicht unterstützt, muss dieser erweitert werden, um ihn USB-fähig zu machen. Dazu müssen die USB-Hostcontroller-Hardware und ein Betriebssystem, das den USB unterstützt, installiert werden. Die Hardware lässt sich über Erweiterungskarten nachrüsten, die in einen PCI-Steckplatz (oder auch in ein Ersatz-Mainboard) eingesetzt werden. Als Windows-Version muss Windows 98 oder einer seiner Nachfolger verwendet werden. Wenn die Hardware die Minimalanforderungen von Windows 98 nicht erfüllt, dann dürften die erforderlichen Aufrüstungen wahrscheinlich mehr als ein neues System mit USB kosten.

Falls es nicht sinnvoll ist, den PC so aufzurüsten, dass er USB unterstützt, dann lässt sich vielleicht noch ein Konverter einsetzen, der zwischen der USB-Schnittstelle des Peripheriegeräts und der RS-232-Schnittstelle, der parallelen Schnittstelle oder einer anderen Schnittstelle vermittelt. Konverter stellen meist aber keine geeignete Lösung dar, wenn der PC eine veraltete Schnittstelle besitzt. Die Entwicklung und Herstellung eines Konverters mit der erforderlichen Hostcontroller-Hardware und dem Code, der normalerweise auf einem PC ausgeführt wird, ist praktisch üblicherweise zu kostspielig.

Doch selbst auf neuen Systemen wollen Anwender gelegentlich Anwendungen unter älteren Betriebssystemen wie etwa MS-DOS ausführen. Nun sind die Softwaretreiber, die von Windows-Anwendungen zur Kommunikation mit USB-Geräten genutzt werden aber größtenteils Windows-spezifisch. Ohne Treiber kann auf USB-Peripheriegeräte jedoch nicht zugegriffen werden. Auch wenn es durchaus möglich ist, USB-Gerätetreiber für DOS zu schreiben, werden solche Treiber von Peripheriegeräte-Herstellern nur höchst selten angeboten. Ausnahmen bilden Maus und Tastatur, da diese vom System-BIOS der meisten Rechner direkt unterstützt werden, um jederzeit einsatzbereit zu sein, d.h. auch in DOS, in den BIOS-Bildschirmen, die beim Hochfahren zu sehen sind, und im geschützten Modus von Windows.

Natürlich verringert sich das Problem der Unterstützung von Legacy-Hardware und älteren Betriebssystemen mit der Zeit, da diese Systeme nach und nach ersetzt werden.

Herausforderungen an die Entwickler

Aus Sicht der Entwickler besteht die wesentliche Herausforderung von USB in der recht komplexen Programmierung und für in geringerem Umfang tätige Entwickler in der Notwendigkeit zur Beschaffung einer Anbieterkennung.

Protokollkomplexität. Ein USB-Peripheriegerät ist intelligent und weiß, wie es auf Anforderungen und andere Ereignisse auf dem Bus reagieren muss. Chips variieren hinsichtlich des Umfangs der zur Durchführung der USB-Kommunikation erforderlichen Firmware-Unterstützung. Um ein USB-Peripheriegerät zu programmieren, muss man meist eine ganze Menge über USB-Protokolle bzw. die Regeln für den Datenaustausch über den Bus wissen. Rechnerseitig wird durch den Gerätetreiber vermieden, dass Anwendungsprogrammierer allzu viele Einzelheiten wissen müssen; allerdings müssen dafür Entwickler von Gerätetreibern mit USB-Protokollen und den Aufgaben der Treiber vertraut sein.

Im Unterschied dazu können einige ältere Schnittstellen mit einfachen Mitteln für andere Anwendungen umfunktioniert werden. So enthält z.B. der Original-Paralleldruckerport des PC nicht mehr als eine Reihe digitaler Ein- und Ausgänge. An ihn können einfache Ein- und Ausgabe-Schaltungen wie etwa Relais, Schalter und

Analog-Digital-Umsetzer angeschlossen werden, ohne dass auf Seiten des Peripheriegeräts irgendwelche Computerintelligenz erforderlich wäre. Die PC-Software kann die einzelnen Bits an den Ports überwachen und steuern.

Bei USB können Anwendungen nicht einfach Daten in Port-Adressen schreiben oder aus diesen auslesen, und Geräten kann man nicht einfach eine Reihe von E/A-Adressen übergeben, die zum direkten Schreiben oder Lesen von Daten genutzt werden können. Um auf ein USB-Gerät zuzugreifen, müssen Anwendungen mit einem Klassen- oder Gerätetreiber kommunizieren, der wiederum auf niedrigerer Ebene mit den USB-Treibern kommuniziert, die die Kommunikation über den Bus steuern und verwalten. Im Gerät müssen die Protokolle implementiert sein, mit deren Hilfe der PC das Gerät erkennen und identifizieren und mit ihm kommunizieren kann.

Zunehmende Betriebssystemunterstützung. Windows enthält Klassentreiber, über die Anwendungen mit vielen Geräten kommunizieren können. Häufig lassen sich Geräte so entwickeln, dass sie von den Klassentreibern unterstützt werden. Falls nicht, können Sie möglicherweise aber einen vom Hersteller des Controllerchips gelieferten Treiber verwenden oder anpassen. Wenn Sie eigene Treiber erstellen müssen, dann können Sie Toolkits verwenden, die Ihnen das Schreiben der USB-Treiber erleichtern.

Gebühren. Das USB-IF stellt die USB-Spezifikationen, verwandte Dokumente, Software für Kompatibilitätstests und mehr online auf seiner Website kostenlos zur Verfügung. Jeder kann USB-Software entwickeln, ohne dass Lizenzgebühren gezahlt werden müssen.

Allerdings muss jeder, der ein Gerät mit einer USB-Schnittstelle vertreibt, beim Forum das Recht zur Benutzung einer Anbieter-ID erwerben. Die dabei zu entrichtende Verwaltungsgebühr beträgt derzeit 1500 Dollar. Alternativ kann für 2500 Dollar jährlich eine USB-IF-Mitgliedschaft erworben werden, die neben der ID noch andere Vorzüge bietet wie z.B. die Teilnahme an Workshops. Die Anbieter-ID und eine vom Anbieter bzw. Hersteller zugeordnete Produkt-ID werden in das Produkt integriert, so dass es eindeutig vom Betriebssystem erkannt werden kann.

Die Gebühren stellen bei höheren Stückzahlen eines Produkts kein Problem dar, können bei Entwicklungen, die nur in kleinen Stückzahlen und bei niedrigem Produktpreis vertrieben werden, jedoch ein Hindernis sein. Bei einigen Controllern, die den Treiber des Herstellers verwenden und keine Programmierung der USB-Schnittstelle seitens des Anbieters erfordern, können Entwickler von Peripheriegeräten die Anbieterkennung des Chipherstellers und eine Produktkennung verwenden, die ihnen vom Chiphersteller zugeteilt wurde.

1.2 Evolution einer Schnittstelle

Der Hauptgrund, weshalb sich neue Schnittstellen nur selten durchsetzen können, besteht darin, dass die Peripheriegeräte, die die Anwender nicht aufgeben möchten, eine unwiderstehliche Anziehungskraft auf die vorhandenen Schnittstellen ausüben. Die Nutzung einer vorhandenen Schnittstelle spart zudem die für die Entwicklung einer neuen Schnittstelle erforderliche Zeit und Kosten. Dies war es auch, warum die Entwickler des ursprünglichen IBM PC die Kompatibilität mit der vorhandenen parallelen Centronics-Schnittstelle und mit der seriellen RS-232-Schnittstelle wählten. Sie wollten den Entwicklungsprozess beschleunigen und es ermöglichen, dass die Anwender die bereits auf dem Markt angebotenen Drucker und Modems anschließen können. Diese Schnittstellen verrichteten fast zwei Jahrzehnte lang ihren Dienst. Dennoch wurden sie mit wachsender Computerleistung und Anzahl und Vielfalt der Peripheriegeräte immer mehr zu einem Engpass der Kommunikation mit zudem nur begrenzten Erweiterungsmöglichkeiten.

Ein Bruch mit der Tradition ist gerechtfertigt, wenn der Nutzen der Verbesserungen die durch die Änderung verursachten Unannehmlichkeiten und Kosten überwiegen. Eine solche Situation führte auch zur Entwicklung des USB.

Das Urheberrecht an der USB-Spezifikation 2.0 liegt gemeinsam bei sieben Unternehmen, die sich alle intensiv mit der Entwicklung von PC-Hardware und -Software befassen: Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC und Philips. Auf der Website des USB-IF finden Sie die USB-Spezifikation 2.0, verwandte Dokumente und weitere Informationen sowohl für Entwickler als auch für Endanwender.

1.2.1 Der ursprüngliche USB

Version 1.0 der USB-Spezifikation wurde im Januar 1996 veröffentlicht. Version 1.1 datiert vom September 1998. Mit USB 1.1 kam ein neuer Transfertyp hinzu (Interrupt OUT). In diesem Buch bezieht sich 1.x auf USB 1.0 und 1.1. Im April 2000 wurde USB 2.0 mit der zusätzlichen High-Speed-Option veröffentlicht. Im Dezember 2000 wurden Korrekturen veröffentlicht und ein neuer Mini-B-Anschluss definiert. ECNs (Engineering Change Notices – Änderungsmitteilung für Entwickler) enthalten Revisionen und Zusätze zur Spezifikation wie z.B. die Definition des Mini-B-Anschlusses, Spezifikationen für Geräte mit Pull-up- und Pull-down-Widerständen mit größeren Toleranzen und die Definition eines neuen Deskriptortyps (der Interface-Association-Deskriptor).

Im PC wurde USB erstmals mit dem OEM Service Release 2 von Windows 95 verfügbar, das nicht direkt für Endkunden erhältlich war. Diese Version stand nur Händlern zur Verfügung, die auf den von ihnen angebotenen PCs Windows 95 installierten. Die USB-Unterstützung war in dieser Version noch fehlerhaft und

eingeschränkt, und außerdem waren nur wenige USB-Peripheriegeräte erhältlich, so dass der Nutzen von USB noch recht beschränkt war.

Besser wurde es mit der Veröffentlichung von Windows 98 im Juni 1998. Damals wurden bereits wesentlich mehr USB-Peripheriegeräte im Handel angeboten, und der USB begann als Schnittstelle Fuß zu fassen. Mit der zweiten Ausgabe von Windows 98 (Windows 98 SE) wurden einige Fehler korrigiert und die USB-Unterstützung weiter verbessert. Um die ursprüngliche Windows 98-Version von Windows SE zu unterscheiden, wird erstere mittlerweile auch Windows 98 Gold genannt.

Dieses Buch konzentriert sich auf Rechner, auf denen Windows 98 oder höher läuft. Windows NT 4 wurde vor Windows 98 veröffentlicht und enthielt noch keine USB-Unterstützung. Windows 2000, Windows Me, Windows XP und Windows Server 2003 bieten durchweg USB-Unterstützung.

Der Begriff *PC* soll in diesem Buch alle verschiedenen Computer umfassen, deren gemeinsamer Urahne der Original-IBM PC ist. Mit Windows 98 oder höher sind hier Windows 98, Windows 98 SE, Windows 2000, Windows Me, Windows XP, Windows Server 2003 und wahrscheinlich auch alle nachfolgenden Windows-Versionen gemeint. Auf einem USB-fähigen PC muss Windows 98 oder höher laufen. Bei einem Host-Computer handelt es sich um einen beliebigen Rechner, der mit USB-Peripheriegeräten kommunizieren kann.

1.2.2 USB 2.0

Als USB 1.x an Beliebtheit gewann, wurde schnell klar, dass eine höhere Busgeschwindigkeit nützlich wäre. Untersuchungen zeigten, dass 40fach höhere Busgeschwindigkeiten als im Full-Speed-Modus mit den Low-Speed- und Full-Speed-Schnittstellen abwärtskompatibel bleiben konnten. Version 2.0 unterstützt eine Busgeschwindigkeit von 480 MBit/s und sorgt dafür, dass USB für Peripheriegeräte wie Drucker, Scanner, Festplatten und Videoanwendungen wesentlich attraktiver ist.

Ein externer USB-2.0-Hub muss alle drei Geschwindigkeiten unterstützen. Andere USB-2.0-Geräte können Low-Speed, Full-Speed, High-Speed oder auch mehrere dieser Modi unterstützen. Der USB 2.0 ist abwärtskompatibel zu USB 1.1. Anders gesagt nutzen Peripheriegeräte für die Version 2.0 die gleichen Steckverbinder und Kabel wie 1.x-Peripheriegeräte, und ein USB-2.0-Peripheriegerät funktioniert, wenn es an einen PC angeschlossen wird, der USB 1.x oder 2.0 unterstützt. Um den High-Speed-Modus nutzen zu können, muss ein High-Speed-fähiges Gerät an einen 2.0-Host-Computer angeschlossen werden und alle Hubs zwischen dem Gerät und dem Host müssen 2.0-Hubs sein. 2.0-Hosts und 2.0-Hubs können auch mit 1.x-Peripheriegeräten kommunizieren. Ein 2.0-kompatibler Hub, an den ein langsames Gerät angeschlossen ist, wandelt die Geschwindigkeiten bei Bedarf um. Zwar werden 2.0-Hubs zwar komplizierter, gleichzeitig wird aber Busband-

breite erhalten und es entfällt die Notwendigkeit unterschiedlicher Hubs für die verschiedenen Geschwindigkeiten.

Als USB-2.0-Geräte anfangs verfügbar wurden, bestand unter den Anwendern Verunsicherung hinsichtlich der Frage, ob alle USB-2.0-Geräte den High-Speed-Modus unterstützen. Das USB-IF versuchte die Verwirrung dadurch zu verringern, dass es Empfehlungen für Bezeichnungen und die Verpackung veröffentlichte, die Geschwindigkeit und Kompatibilität und nicht USB-Versionsnummern hervorhoben. Die Empfehlungen besagen, dass ein Produkt, das den High-Speed-Modus unterstützt, als »Hi-Speed USB« gekennzeichnet werden sollte, wobei zu den Angaben auf der Verpackung »Voll kompatibel mit Original-USB« und »Kompatibel mit der USB-2.0-Spezifikation« zählen sollten. Ein Produkt, das nur Low-Speed und Full-Speed unterstützt, ist nur ein »USB«-Produkt, und die empfohlenen Angaben auf der Verpackung sollten »Kompatibel mit der USB-2.0-Spezifikation« und »Arbeitet mit USB und Hi-Speed-USB-Systemen, -Peripheriegeräten und Kabeln zusammen«. Hersteller sollten Bezugnahmen auf Low-Speed oder Full-Speed auf Verpackungen für Endverbraucher vermeiden.

1.2.3 USB On-The-Go

Als USB für alle Arten von Peripheriegeräten zur Schnittstelle der Wahl geworden war, fragten Entwickler nach Möglichkeiten zur direkten Verbindung ihrer Peripheriegeräte miteinander und mit anderen USB-Peripheriegeräten. Ein Anwender kann z.B. einen Drucker direkt an eine Kamera anschließen oder zwei Festplatten für den Austausch von Dateien miteinander verbinden wollen. Die 2001 veröffentlichte OTG-Ergänzung (On-The-Go) zur USB-2.0-Spezifikation definiert eine Host-Funktion mit eingeschränkten Fähigkeiten, die in Geräte implementiert werden kann, um die Kommunikation mit Peripheriegeräten zu ermöglichen.

1.2.4 Wireless USB

Bei einer weiteren USB-Erweiterung handelt es sich um den *Wireless Universal Serial Bus*. Diese Spezifikation wurde im Mai 2005 veröffentlicht und ist über das USB-IF erhältlich. Sie ermöglicht die drahtlose Kommunikation der Geräte bei einer Bruttotransferrate von maximal 480 MBit/s.

Wireless-USB-Geräte müssen Datentransferraten von 53,3, 106,7 und 200 MBit/s unterstützen. Optional können die verbleibenden Datenraten der Spezifikation von 80, 160, 320, 400 und 480 MBit/s unterstützt werden.

1.2.5 USB und IEEE-1394 im Vergleich

Eine weitere verbreitete Schnittstelle für neue Peripheriegeräte ist die IEEE-1394. Apples Computerrealisierung der Schnittstelle heißt FireWire. Allgemein ist IEEE-1394 zwar schneller und flexibler als USB, aber auch teurer.

Bei USB steuert ein einzelner Host die Kommunikation mit vielen Geräten. Den kompliziertesten Teil der Bearbeitung übernimmt der Host, so dass die Elektronik der Geräte verhältnismäßig einfach und preiswert sein kann. IEEE-1394-Geräte können direkt miteinander kommunizieren, und einzelne Kommunikationsvorgänge können auch für mehrere Empfänger bestimmt sein. Daraus resultiert eine flexiblere Schnittstelle, bei der aber die Geräteelektronik komplizierter und teurer ist.

IEEE-1394 eignet sich am besten für Anwendungen mit extrem schneller Datenübertragung oder bei denen Daten an mehrere Empfänger versandt werden müssen. USB eignet sich am besten für verbreitete Peripheriegeräte wie Tastatur, Drucker und Scanner sowie für Anwendungen mit geringen bis moderaten Geschwindigkeitsanforderungen und Anwendungen, bei denen die Kosten eine vorrangige Rolle spielen. Für viele Geräte wie z.B. Laufwerke, sind beide Schnittstellen gleichermaßen geeignet. Und tatsächlich enthalten einige Geräte beide Schnittstellen.

1.2.6 USB und Ethernet im Vergleich

Bei einigen Anwendungen stehen USB und Ethernet zur Auswahl. Zu den Vorteilen von Ethernet zählen die Benutzbarkeit sehr langer Kabel, die Möglichkeit des gleichzeitigen Versendens von Mitteilungen an mehrere Empfänger (broadcasting) und die Unterstützung von Internet-Protokollen in PCs und Ethernet-fähigen Entwicklungssystemen. Wie IEEE-1394 ist jedoch die zur Unterstützung von Ethernet erforderliche Hardware komplexer und teurer als die für USB-Peripheriegeräte typische Hardware. USB ist mit seinen vier Transfertype und einer Vielzahl für verschiedene Zwecke definierte Klassen auch vielseitiger.

1.3 Bus-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 PC oder ein anderer Computer, der einen Hostcontroller und einen Root-Hub (Stammhub) enthält. Diese Komponenten arbeiten zusammen, so dass das Betriebssystem mit den Geräten am Bus kommunizieren kann. Der Hostcontroller formatiert die Daten zur Übertragung auf dem Bus und übersetzt die empfangenen Daten in ein Format, das die Betriebssystemkomponenten verstehen können. Daneben 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. Der Root-Hub erkennt zusammen mit dem Hostcontroller, wenn Geräte angeschlossen oder entfernt werden, führt

Anforderungen des Hostcontrollers aus und überträgt Daten zwischen Hostcontroller und Geräten.

Bei den Geräten kann es sich sowohl um Peripheriegeräte als auch um zusätzlich an den Bus angeschlossene Hubs handeln. Ein Hub weist einen oder mehrere Ports zum Anschluss von Geräten auf. In der USB-Spezifikation sind die Kabel und Stecker definiert, über die Geräte und Hubs miteinander verbunden werden.

1.3.1 Die Topologie

Die Topologie, d.h. die Anordnung der Verbindungen am Bus, ist eine kaskadierte Stern-Topologie (siehe Abbildung 1.1). Im Zentrum jedes Sterns befindet sich ein Hub. Jeder Punkt an einem Stern ist ein Gerät, das an einen der Ports an einem Hub angeschlossen ist. 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.

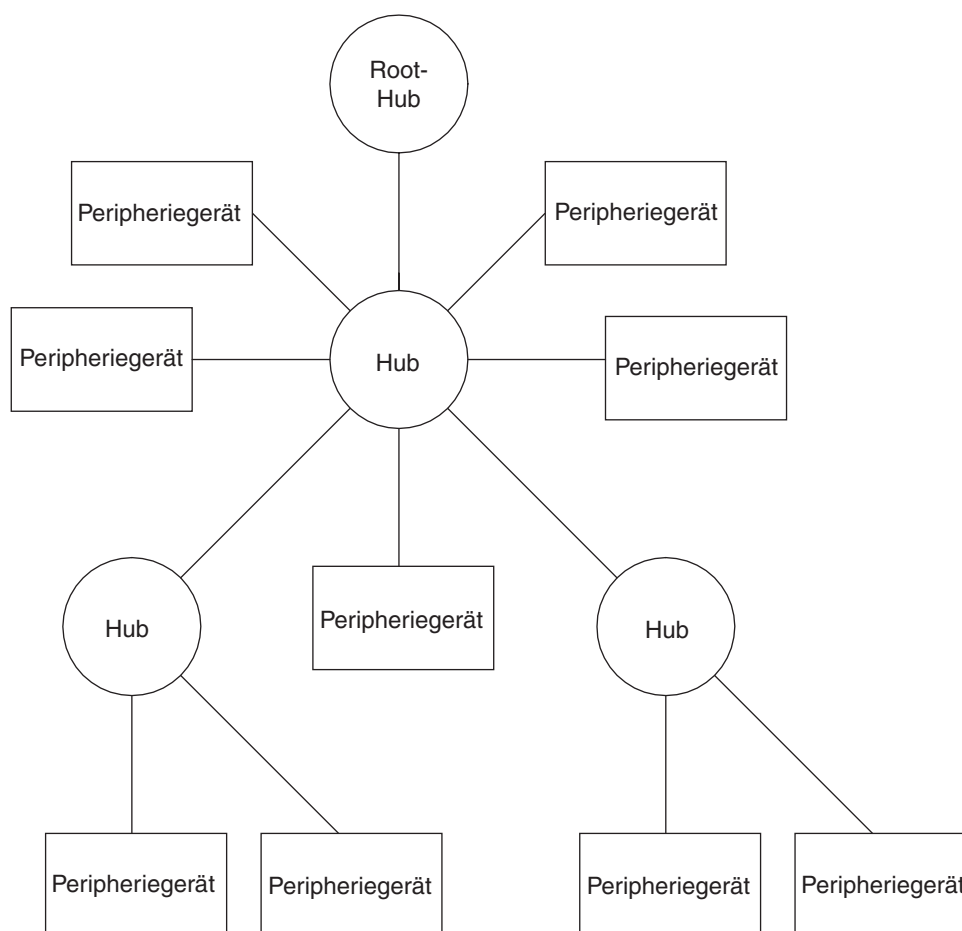


Abb. 1.1: Der USB nutzt eine kaskadierte Stern-Topologie, bei der jeder Hub das Zentrum eines Sterns ist, der selbst an Peripheriegeräte oder an zusätzliche Hubs angeschlossen sein kann.

Der kaskadierte Stern beschreibt nur die physischen Verbindungen. Bei der Programmierung spielt lediglich die logische Verbindung eine Rolle. Um miteinander zu kommunizieren, müssen der Host und das Gerät nicht wissen, über wie viele Hubs die Kommunikation abgewickelt wird.

Es kann jeweils immer nur ein Gerät mit einem Hostcontroller kommunizieren. Um die für USB-Geräte verfügbare Bandbreite zu erhöhen, kann ein PC über mehrere Hostcontroller verfügen.

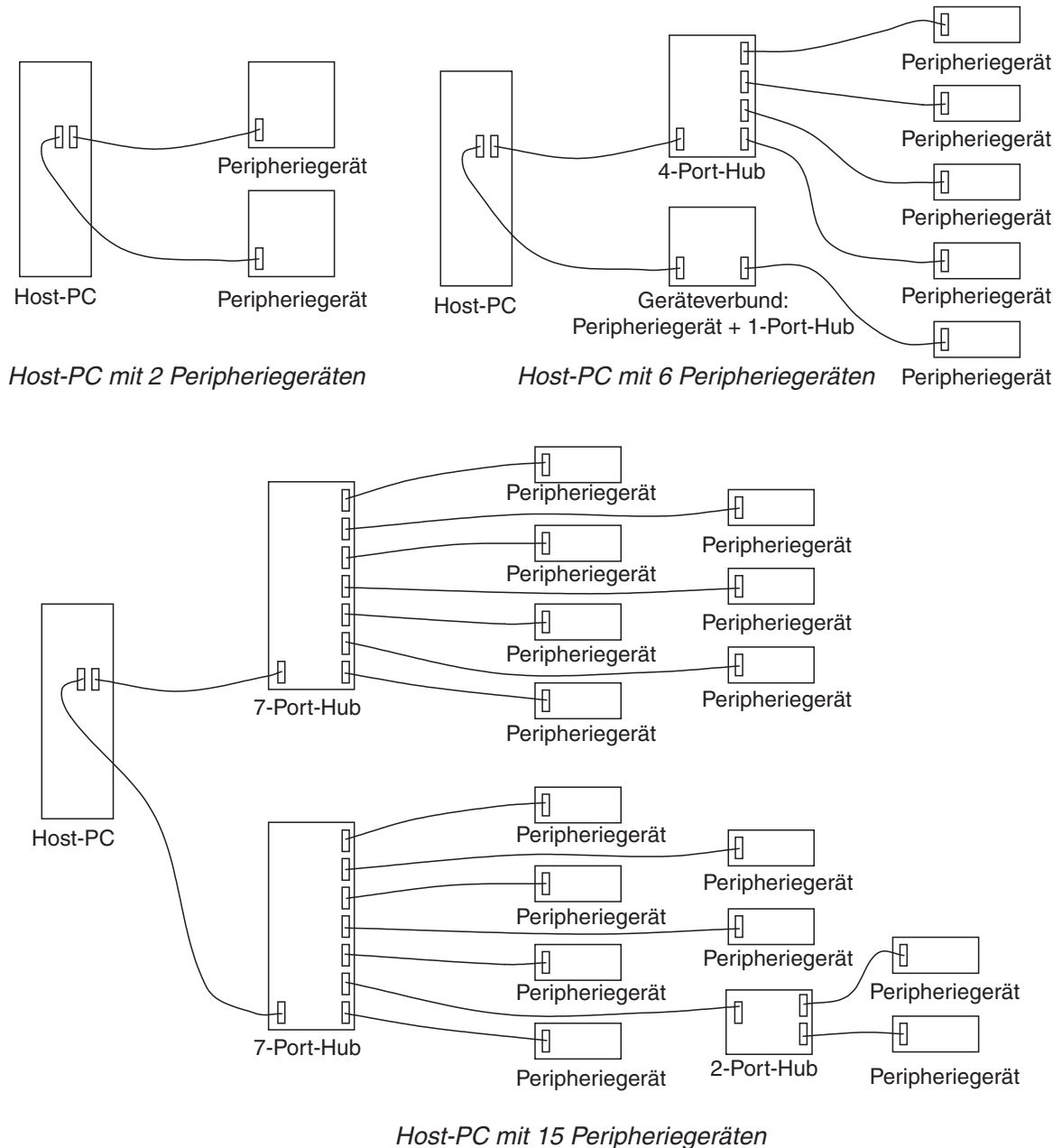


Abb. 1.2: Es gibt viele Möglichkeiten des Anschlusses von USB-Geräten an einen Host-PC. Die Abbildung zeigt einige der Optionen für einen Host mit zwei Ports.

Abbildung 1.2 zeigt einige der möglichen Konfigurationen für einen PC mit zwei USB-Anschlüssen. Einige Geräte sind Verbundgeräte (composite oder compound devices), die sowohl ein Peripheriegerät als auch einen Hub enthalten. Man kann bis zu fünf Hubs und bis zu insgesamt 127 Peripheriegeräte und Hubs (einschließlich des Root-Hubs) hintereinander schalten. Es kann jedoch unzweckmäßig sein, wenn so viele Geräte einen gemeinsamen Datenpfad zur Kommunikation mit einem einzelnen Hostcontroller benutzen.

In einigen Fällen, vor allem bei Verbundgeräten, bei denen die Hubs im Peripheriegerät verborgen sind, könnte der Eindruck entstehen, dass die Peripheriegeräte hintereinander geschaltet sind und jedes neue Peripheriegerät mit dem letzten in einer Kette verbunden ist. Aber die USB-Topologie ist sowohl flexibler als auch komplizierter als einfach verkettete Geräte. Zwar muss jedes Peripheriegerät an einen Hub angeschlossen werden, aber die Peripheriegeräte und die Hubs brauchen nicht unbedingt in einer einzigen Kette verbunden zu sein.

1.3.2 Begriffsbestimmungen

In der USB-Welt besitzen einige alltägliche Begriffe spezifische Bedeutungen. Zusammen mit dem *Host*, den wir zuvor als Computer definiert hatten, der die Schnittstelle steuert, sind drei weitere derartige Begriffe *Funktion*, *Hub* und *Gerät*. Ebenfalls wichtig ist es, das Konzept des USB-Ports zu verstehen und die Unterschiede zu anderen Ports bzw. Anschlüssen, wie z.B. RS-232, zu kennen.

Funktion

Die USB-Spezifikation definiert eine *Funktion* als ein Gerät, das für den Host eine einzelne Fähigkeit bereitstellt. Eine Maus, ein Lautsprecherpaar oder eine Datenerfassungseinheit sind Beispiele für eine Funktion. Ein einzelnes physisches Gerät kann mehr als eine Funktion enthalten.

Hub

Ein *Hub* besitzt einen Upstream-Anschluss, der der Kommunikation mit dem Host dient, und einen oder mehrere Downstream-Anschlüsse oder interne Verbindungen zu eingebetteten Geräten. Jeder Downstream-Anschluss und jede interne Verbindung ist ein USB-Port.

Ein 1.x-Hub wiederholt die empfangenen USB-Daten in beiden Richtungen, verwaltet die Stromversorgung und sendet und empfängt Status- und Steuermeldungen. Ein 2.0-Hub kann all dies und unterstützt darüber hinaus den High-Speed-Modus und wandelt bei Bedarf die Signale für die verschiedenen Geschwindigkeiten um.

Gerät

Die Definition der USB-Spezifikation für ein *Gerät* lautet, dass es sich um eine Funktion oder einen Hub handelt, wobei es den Spezialfall des *Verbundgeräts* gibt, das sowohl einen Hub als auch eine oder mehrere Funktionen enthält. Der Host behandelt ein Verbundgerät weitgehend so, als ob es sich bei dem integrierten Hub und den Funktionen des Verbundgeräts um eigenständige physische Geräte handeln würde. Alle Geräte besitzen eindeutige Adressen am Bus, wobei auch hier Verbundgeräte eine Ausnahme darstellen, da hier sowohl Hub als auch Funktionen jeweils eindeutige Adressen besitzen.

Ein *Verbundgerät* ist ein multifunktionales Gerät mit mehreren unabhängigen Schnittstellen. Die Schnittstellen werden von den im Gerät gespeicherten Schnittstellen-Deskriptoren beschrieben. Ein Verbundgerät besitzt eine Adresse am Bus, aber die jeweiligen Schnittstellen besitzen unterschiedliche Funktionen und benötigen unterschiedliche Gerätetreiber auf dem Host. So kann ein Verbundgerät z.B. eine Schnittstelle für ein Audiogerät und eine andere für ein Bedienfeld besitzen. In der Microsoft-Dokumentation wird der Begriff Verbundgerät (composite device) teilweise und unabhängig davon, ob mehr als eine aktive Schnittstelle vorhanden ist, für beliebige Geräte verwendet, deren Funktion über ihren *interface*-Deskriptor und nicht ihren *device*-Deskriptor definiert werden.

Port

Allgemein ist ein Computerport ein adressierbarer Ort, an den zusätzliche Schaltungen angeschlossen werden können. Üblicherweise schließen diese Schaltkreise mit einem Stecker oder einer Buchse ab, so dass ein Kabel an ein Peripheriegerät angeschlossen werden kann. In einigen Fällen sind die Peripherieschaltkreise fest mit einem Port verdrahtet. Über Software können Daten von den Port-Adresse(n) ausgelesen oder in sie geschrieben werden, so dass die Port-Schaltkreise überwacht und gesteuert werden können. In diesem Zusammenhang werden anstelle des Begriffs »Port« häufig auch »Schnittstelle« oder »Anschluss« verwendet. Auch der Arbeitsspeicher eines Computers besteht aus adressierbaren Speicherzellen, auf die aber mit einem anderen Satz von Maschinenbefehlen als auf Ports zugegriffen wird.

USB-Ports unterscheiden sich von vielen anderen Ports dadurch, dass sämtliche Ports an einem Bus einen gemeinsamen Pfad zum Host nutzen und nicht direkt adressierbar sind. Bei der RS-232-Schnittstelle befindet sich jeder Port im PC und ist unabhängig von den anderen. Falls zwei RS-232-Ports vorhanden sind, besitzt jeder seinen eigenen Datenpfad, wobei jedes Kabel nur seine eigenen Daten und nicht die eines anderen Ports überträgt. Die beiden Ports können hier gleichzeitig Daten senden und empfangen.

Bei USB verwaltet jeder Hostcontroller einen einzelnen Bus bzw. Datenpfad. Jeder Anschluss an einem Bus stellt einen USB-Port dar, wobei aber anders als bei RS-

232-Ports alle Geräte die verfügbare Bus-Bandbreite gemeinsam nutzen. Somit kann ein USB-Hostcontroller zwar mit mehreren Ports kommunizieren, die jeweils über eigene Steckverbinder und Kabel verfügen, aber es existiert dennoch nur ein einziger Datenpfad. Es kann jeweils immer nur ein Gerät oder der Host Daten senden. Ein einzelner Computer kann aber über mehrere USB-Hostcontroller verfügen, die jeweils einen eigenen Bus besitzen. Andere Schnittstellen, bei denen mehrere Geräte einen gemeinsamen Datenpfad nutzen können, sind IEEE-1394, SCSI und Ethernet.

Ein weiterer Unterschied bei USB besteht darin, dass ein Bus über Ports verfügen kann, die sich nicht am Hostcontroller im PC befinden.

1.4 Arbeitsteilung

Die Aufgaben für den Host und seine Geräte sind unterschiedlich definiert. Der Host übernimmt einen Großteil der Aufgabe der Verwaltung der Kommunikation. Die Geräte müssen aber intelligent genug sein, um auf die Kommunikation und andere Busereignisse seitens des Hosts und der Hubs, an die sie angeschlossen sind, reagieren zu können.

1.4.1 Aufgaben des Hosts

Um mit USB-Geräten kommunizieren zu können, benötigt ein Computer Hardware- und Software-Unterstützung, um als USB-Host fungieren zu können. Die Hardware besteht aus einem USB-Hostcontroller und einem Root-Hub mit einem oder mehreren USB-Ports. Für die Softwareunterstützung sorgt ein Betriebssystem, das Mechanismen enthält, über die Treiber mit der USB-Hardware kommunizieren können.

Nahezu jeder moderne PC verfügt über einen USB-Hostcontroller und zwei oder mehr USB-Port-Anschlüsse. Viele PCs enthalten mehrere Hostcontroller. Wenn in das Mainboard eines Computers keine USB-Unterstützung integriert ist, dann können Sie einen Hostcontroller auf einer Erweiterungskarte nachrüsten, die in einen Steckplatz für den PCI-Bus eingesetzt wird. Für portable Rechner sind USB-Controller in PC-Cards erhältlich.

Der Host ist für den Bus verantwortlich. Der Host 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. An einen Bus können zahlreiche Geräte jeweils mit ganz unterschiedlichen Eigenschaften angeschlossen sein, die alle gleichzeitig Daten übertragen möchten. Damit ist die Aufgabe des Hosts also durchaus nicht trivial!

Zum Glück wird die meiste Arbeit zur Verwaltung des Busses von der Hardware des Hostcontrollers und den Treibern des Hostcontrollers in Windows übernommen. Für jedes an den Host angeschlossene Gerät wird zudem ein Gerätetreiber benötigt, der die Kommunikation von Anwendungsprogrammen mit dem Gerät ermöglicht. Einige Peripheriegeräte können zusammen mit Windows gelieferte Gerätetreiber benutzen. Andere Geräte benutzen Treiber, die von den Geräteherstellern zur Verfügung gestellt werden. Verschiedene Softwarekomponenten auf Systemebene steuern die Kommunikation zwischen dem Gerätetreiber, dem Hostcontroller und der Hardware des Root-Hubs.

Anwendungsprogramme brauchen sich nicht um die USB-spezifischen Details der Kommunikation mit den Geräten zu befassen. Sie müssen zum Senden und Empfangen von Daten nur auf Standardfunktionen des Betriebssystems zurückgreifen, die mit nahezu allen Programmiersprachen angesprochen werden können. Häufig muss die Anwendung nicht einmal wissen, ob ein Gerät den USB oder eine andere Schnittstelle verwendet.

Der Host übernimmt die nachfolgend aufgeführten Aufgaben. Dabei handelt es sich um allgemein gehaltene Beschreibungen. In den nachfolgenden Kapiteln erfahren Sie dann die Einzelheiten.

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. Immer wenn nach dem Einschalten ein Gerät entfernt oder angeschlossen wird, erfährt der Host jeweils von diesem Ereignis, wobei er neu angeschlossene Geräte enumeriert und abgetrennte Komponenten aus seiner Liste der für Anwendungen verfügbaren Geräte entfernt.

Verwaltung des Datenflusses

Der Host verwaltet den Datenfluss auf dem Bus. Es ist möglich, dass mehrere Peripheriegeräte gleichzeitig Daten senden wollen. Der Hostcontroller unterteilt die verfügbare Zeit in Segmente, die Frames und Mikroframes genannt werden, und stellt jeder Übertragung Teile eines Frames oder Mikroframes zur Verfügung.

Übertragungen, die mit einer garantierten Rate stattfinden müssen, wird in jedem Frame die benötigte Zeit zugeteilt. Während der Enumeration fordert ein Gerätetreiber die Bandbreite an, die für Transfers mit garantiertem Zeitbedarf benötigt wird. Wenn die benötigte Bandbreite nicht verfügbar ist, unterbindet der Host die Kommunikation mit dem Gerät. Dann kann der Treiber einen kleineren Teil der Bandbreite anfordern oder warten, bis die angeforderte Bandbreite verfügbar ist. Übertragungen ohne garantierten Zeitbedarf verwenden den verbleibenden Teil der Frames und müssen möglicherweise warten, wenn der Bus ausgelastet ist.

Fehlerprüfung

Beim Übertragen von Daten fügt der Host den übertragenen Daten Fehlerprüfbits hinzu. Beim Empfangen von Daten berechnet das Gerät eine Prüfsumme für die Daten und vergleicht diese mit den empfangenen Fehlerprüfbits. Wenn das Ergebnis nicht übereinstimmt, bestätigt das Gerät den Empfang der Daten nicht, so dass der Host weiß, dass sie erneut übertragen werden sollten. (Daneben unterstützt USB aber auch einen Transfertyp, der im Interesse der Gewährleistung einer konstanten Übertragungsrates keine erneute Übertragung zulässt.) Auf ähnliche Weise prüft der Host die von Geräten empfangenen Daten auf Fehler.

Der Host kann andere Hinweise empfangen, wenn ein Gerät keine Daten senden oder empfangen kann. Er kann daraufhin den Treiber des Geräts über das Problem informieren, der seinerseits die Anwendung benachrichtigen kann, so dass sie geeignete Maßnahmen ergreift.

Stromversorgung

Neben seinen zwei Signalleitungen besitzt das USB-Kabel eine +5-V- und Masseleitungen. Einige Geräte können ihren gesamten Strombedarf über diese Leitungen decken. Der Host liefert beim Einschalten oder Anschließen Strom an alle Geräte, wobei er mit ihnen zusammenarbeitet, um so weit wie möglich Strom zu sparen. Jedes Bus-betriebene Gerät mit hoher Energieaufnahme kann je Bussegment, das mit Strom versorgt wird, maximal 500 Milliampere über den Bus beziehen. Die Ports an einigen batteriebetriebenen PCs und Hubs unterstützen nur Geräte mit niedrigem Stromverbrauch, die höchstens 100 Milliampere benötigen dürfen. Andererseits können USB-Geräte auch über ihre eigene Stromversorgung verfügen.

Datenaustausch mit Peripheriegeräten

Alle oben genannten Aufgaben unterstützen den Datenaustausch mit Peripheriegeräten und damit die Hauptaufgabe des Hosts. In einigen Fällen verlangt ein Gerätetreiber, dass der Host Daten in festen Intervallen Daten zu senden oder zu empfangen versucht, während der Host in anderen Fällen nur dann mit den Geräten kommuniziert, wenn Anwendungen oder andere Softwarekomponenten einen Transfer einzuleiten versuchen. Der Gerätetreiber informiert dann die entsprechenden Anwendungen über gegebenenfalls auftretende Probleme.

1.4.2 Aufgaben des Peripheriegeräts

Die Aufgaben des Peripheriegeräts (oder einfach Geräts) stellen in vielerlei Hinsicht ein Spiegelbild der Host-Aufgaben dar. Wenn der Host einen Kommunikationsvorgang initiiert, muss das Peripheriegerät antworten. Allerdings besitzen Peripheriegeräte auch spezifische Aufgaben. Ein Gerät kann von selbst keine USB-Kommunikation beginnen. Stattdessen muss es warten und auf Kommunikations-

anforderungen des Hosts antworten (eine Ausnahme ist die Remote-Wakeup-Funktion, über die ein Gerät den Host zur Kommunikation auffordern kann).

Die Hardware des USB-Controllers im Peripheriegerät führt viele der Aufgaben des Geräts automatisch aus. In welchem Ausmaß die Geräte-Firmware Unterstützung bieten muss, ist von Chip zu Chip verschieden.

Das Peripheriegerät muss alle nachfolgend beschriebenen Aufgaben übernehmen. Hierbei handelt es sich um eine allgemeine Beschreibung. Ausführlichere Informationen finden Sie in den nachfolgenden Kapiteln.

Erkennen von Kommunikation, die für den Chip bestimmt ist

Jedes Gerät überwacht die in jeder Kommunikationsaktivität über den Bus enthaltene Geräteadresse. Wenn die Adresse nicht mit der im Gerät gespeicherten Adresse übereinstimmt, ignoriert es den Kommunikationsvorgang. Wenn die Adresse übereinstimmt, speichert das Gerät die Daten in seinem Empfangspuffer und löst einen Interrupt aus, um zu signalisieren, dass Daten eingetroffen sind. Diese Funktionen sind bei fast allen Chips in deren Hardware integriert, so dass sie vom Programmcode nicht unterstützt werden müssen. Bis der Chip einen Kommunikationsvorgang mit seiner Geräteadresse erkannt hat, braucht die Firmware des Geräts keinerlei Aktion auszuführen oder Entscheidungen zu fällen.

Reaktionen auf Standardanforderungen

Beim Einschalten oder Anschließen des Geräts an ein eingeschaltetes System muss ein Gerät die während der Enumeration vom Host gesendeten Standardanforderungen beantworten. Auch nach abgeschlossener Enumeration kann der Host jederzeit Standardanforderungen übertragen.

Alle Geräte müssen auf diese Anforderungen reagieren, die die Fähigkeiten und den Status des Geräts abfragen oder das Gerät zu anderen Aktionen auffordert. Beim Empfang einer Anforderung legt das Gerät die als Antwort an den Host zu sendenden Daten oder Statusinformationen in einem Sendepuffer ab. Bei einigen Anforderungen wie etwa bei der Auswahl einer Konfiguration sendet das Gerät nicht nur Daten, sondern ergreift darüber hinaus weitere Maßnahmen.

Die USB-Spezifikation definiert elf Anforderungen. In Klassen oder seitens des Herstellers können weitere Anforderungen definiert werden. Allerdings braucht ein Gerät nicht alle Anforderungen auszuführen; es muss lediglich verständlich auf Anforderungen reagieren. Wenn der Host z.B. eine vom Gerät nicht unterstützte Konfiguration anfordert, dann antwortet das Gerät mit einem Code, der den Host darüber informiert, dass die Konfiguration nicht unterstützt wird.

Fehlerprüfung

Wie der Host ergänzt auch das Gerät zu sendende Daten durch Fehlerprüfbits. Wenn das Gerät Daten mit Fehlerprüfbits empfängt, führt es die Fehlerprüfberech-

nungen aus. Wenn das Gerät antwortet oder nicht antwortet, dann weiß der Host, ob er die Daten erneut senden muss oder nicht. Diese Funktionen sind typischerweise in die Hardware integriert und müssen nicht programmiert werden. Weiterhin erkennt das Gerät auch die vom Host als Antwort auf die von ihm empfangenen Daten übertragenen Empfangsbestätigungen.

Energieverwaltung

Ein Gerät kann Bus-betrieben werden oder ein eigenes Netzteil besitzen. Wenn keine Busaktivität stattfindet, müssen Bus-betriebene Geräte ihre Energieaufnahme über den Bus reduzieren.

Wenn der Host in einen Energie sparenden Status eintritt, wird die gesamte Kommunikation über den Bus einschließlich der periodischen Zeitmarken, die der Host normalerweise sendet, eingestellt. Wenn ein Gerät drei Millisekunden lang keine Busaktivitäten mehr erkennt, dann muss es in den Suspend-Status wechseln und die Stromaufnahme über den Bus beschränken. Während sich das Gerät im Suspend-Status befindet, muss es weiterhin den Bus überwachen und beim Wiederauftreten von Busaktivitäten den Suspend-Status verlassen.

Geräte, die die Remote-Wakeup-Funktion nicht unterstützen, sollten im Suspend-Zustand nicht mehr als maximal 500 Mikroampere über den Bus beziehen. Wenn ein Gerät die Remote-Wakeup-Funktion unterstützt und diese vom Host aktiviert wurde, dann liegt die Grenze bei 2,5 Milliampere. Allerdings handelt es sich dabei um Durchschnittswerte während einer Sekunde, so dass die Spitzenspannung höher liegen kann.

Datenaustausch mit dem Host

Alle oben genannten Aufgaben unterstützen die Hauptaufgabe des USB-Ports des Geräts, den Austausch von Daten mit dem Host. Wenn das Gerät konfiguriert worden ist, muss es auf Kommunikationsanforderungen reagieren, die Daten enthalten könnten und die das Gerät durch Senden von Daten oder Statusinformationen beantworten sollte. Die Fähigkeiten des Geräts, die Gerätetreiber des Hosts und die das Gerät nutzenden Anwendungen bestimmen gemeinsam die Art der Kommunikation und wann diese stattfindet.

Bei den meisten Transfertypen, bei denen der Host Daten zum Gerät überträgt, muss das Gerät jeden Transfer bestätigen. Es sendet dann einen Code, der angibt, ob die Daten akzeptiert wurden oder ob das Gerät zu beschäftigt war, um die Daten zu verarbeiten. Bei den meisten Transfers, bei denen das Gerät Daten zum Host überträgt, muss es auf jeden Kommunikationsversuch reagieren, indem es entweder Daten oder einen Code überträgt, der darauf hinweist, dass entweder keine zu sendenden Daten mehr vorliegen oder dass das Gerät beschäftigt war. Normalerweise übermittelt die Gerätehardware automatisch die passende Antwort. Einige Transfers arbeiten ohne Empfangsbestätigungen. Dann geht der Sender einfach davon aus, dass der Empfänger alle übertragenen Daten empfangen hat.

Die Hardware des Controllerchips übernimmt die Einzelheiten der Formatierung der Daten für den Bus. Die Formatierung umfasst das Hinzufügen von Fehlerprüf-bits zu den zu sendenden Daten, die Fehlerprüfung empfangener Daten und das Senden und Empfangen der einzelnen Bits über den Bus.

Darüber hinaus muss das Gerät natürlich alle weiteren Aufgaben erfüllen, für die es zuständig ist. So muss eine Maus beispielsweise ständig Mausbewegungen und Tastenklicks erfassen, eine Datenerfassungseinheit muss die Daten ihrer Sensoren auslesen und ein Drucker muss die empfangenen Daten zu Papier bringen.

1.4.3 Geschwindigkeit

Ein Controllerchip kann eine oder mehrere Geschwindigkeiten unterstützen. Praktisch alle USB-Hubs unterstützen sowohl Low-Speed- als auch Full-Speed-Geräte. Eine Ausnahme stellen hier allenfalls die Hubs in Verbundgeräten dar, die lediglich eine bestimmte Geschwindigkeit unterstützen. Low-Speed- und Full-Speed-Peripheriegeräte können an beliebige USB-Hubs angeschlossen werden. Da der Benutzer keine Einstellungen oder Konfigurationen für unterschiedliche Geschwindigkeiten wählen muss, braucht er überhaupt nicht zu wissen, ob es sich um Low-Speed- oder Full-Speed-Geräte handelt.

High-Speed-Peripheriegeräte unterstützen wahrscheinlich zwei Geschwindigkeiten, so dass sie auch im Full-Speed-Modus arbeiten können. Ein 1.x-Hub unterstützt keine High-Speed-Transfers, da es den High-Speed-Modus noch gar nicht gab, als die USB-Spezifikation 1.x veröffentlicht wurde. Um zu gewährleisten, dass High-Speed-Geräte 1.x-Hosts und 1.x-Hubs nicht ins Stolpern bringen, müssen alle High-Speed-Geräte im Full-Speed-Modus zumindest die Standardanforderungen zur Enumeration beantworten können. Jeder Host kann dadurch alle angeschlossenen Geräte identifizieren.

High-Speed-Geräte müssen im Full-Speed-Modus aber nur auf das Zurücksetzen des Busses und Standardanforderungen reagieren. Da die Full-Speed-Unterstützung aber leicht implementiert werden kann und sie für das Bestehen der Kompatibilitätstests des USB-IF benötigt wird, sind die meisten High-Speed-Geräte auch im Full-Speed-Modus funktionsfähig.

Die tatsächliche Datenübertragungsrate zwischen Peripheriegerät und Host ist niedriger als die Busgeschwindigkeit und nicht immer voraussagbar. Einige der übertragenen Bits werden für die Identifikation und Synchronisation sowie für die Fehlerprüfung verwendet, so dass die Datenrate auch vom Transfertyp und davon abhängig ist, wie intensiv der Bus anderweitig beansprucht wird.

Für zeitkritische Daten unterstützt USB Transfertypen mit einer garantierten Rate oder garantierter maximaler Latenzzeit. Isochrone Transfers haben eine garantierte Rate, mit der der Host die Übertragung einer spezifischen Anzahl von Bytes in einem festgelegten Intervall anfordern kann. Die Intervalle können bei Full-Speed

eine Millisekunde bzw. bei High-Speed 125 Mikrosekunden betragen. Isochrone Transfers arbeiten aber ohne Fehlerprüfung. Interrupt-Transfers arbeiten mit Fehlerprüfung und garantieren maximale Latenzzeiten. Dabei wird zwar keine genau festgelegte Transferrate garantiert, die Zeitspanne zwischen Transferversuchen wird aber nie größer als ein spezifischer Wert. Dieser Wert kann bei Low-Speed-Geräten zwischen 10 und 255 Millisekunden betragen. Bei Full-Speed-Geräten liegt er zwischen 1 und 255 Millisekunden. Bei High-Speed-Geräten kann der Wert zwischen 125 Mikrosekunden und 4096 Sekunden liegen.

Da der Bus gemeinsam genutzt wird, kann einem Gerät nicht jederzeit eine bestimmte Rate oder maximale Latenzzeit garantiert werden. Wenn der Bus zu stark belegt ist, um eine angeforderte Rate oder Latenzzeit zuzulassen, weigert sich der Host, den Konfigurationsvorgang abzuschließen, der dem Host den Versuch der Datenübertragung ermöglichen würde. Um die reservierte Bandbreite möglichst umfassend nutzen zu können, müssen der Gerätetreiber und die Anwendungsprogramme oder die Geräte-Firmware dafür sorgen, dass die Daten zur Übertragung bereitstehen, wenn der Hostcontroller zur Einleitung des Transfers bereit ist. Der Empfänger der Daten muss auch zur Übernahme der eintreffenden Daten bereit sein.

Im Full-Speed- und im High-Speed-Modus sind Bulk-Transfers die schnellsten Transfers über einen ansonsten untätigen Bus. Mit ihnen lassen sich theoretische Transferraten von maximal 1216 MByte/s im Full-Speed-Modus bzw. 53.248 MByte/s im High-Speed-Modus erreichen. Der Hostcontroller kann einzelne Bulk-Transfers aber auch auf niedrigere Raten beschränken. Die Transfers mit der höchsten garantierten Bandbreite im High-Speed-Modus sind Interrupt-Transfers und isochrone Transfers mit 24.576 MByte/s.

Auch wenn die Low-Speed-Busgeschwindigkeit 1,5 MBit/s beträgt, liegt die höchste garantierte Übertragungsrate für Daten nur bei 8 Bytes in 10 Millisekunden bzw. 800 Byte/s.

1.5 Entwicklung eines Geräts

Die Entwicklung eines USB-Produkts für PCs umfasst die Entwicklung des Peripheriegeräts selbst bis zu seiner Betriebsbereitschaft und die Entwicklung oder Beschaffung der für die Kommunikation mit dem Peripheriegerät benötigten PC-Software.

1.5.1 Elemente der Zusammenarbeit

Für ein USB-Peripheriegerät sind folgende Elemente erforderlich:

- Ein Controllerchip mit USB-Schnittstelle
- Code im Peripheriegerät, der die USB-Kommunikation ausführt

- Die vom Peripheriegerät zur Durchführung seiner sonstigen Funktionen (Datenverarbeitung, Eingaben lesen, Ausgaben schreiben) benötigte Hardware und der dazu erforderliche Programmcode
- Ein Host, der USB unterstützt
- Treibersoftware im Host, damit Anwendungen mit dem Peripheriegerät kommunizieren können
- Wenn das Peripheriegerät keinem der bereits vom Betriebssystem unterstützten Standardtypen entspricht, benötigt der Host weiterhin Anwendungssoftware, damit Anwender auf das Peripheriegerät zugreifen können. Bei Standardperipheriegeräten wie z.B. Maus, Tastatur oder Laufwerk werden keine angepassten Anwendungsprogramme benötigt, auch wenn Sie möglicherweise Testanwendungen schreiben wollen.

1.5.2 Hilfsmittel für die Entwicklung

Zur Entwicklung eines USB-Peripheriegeräts werden folgende Hilfsmittel benötigt:

- Ein Assembler oder Compiler zum Erstellen der Geräte-Firmware (des im Controllerchip ausgeführten Codes).
- Ein Programmiergerät, um den assemblierten oder kompilierten Code im Programmspeicher des Controllers zu speichern
- Eine Programmiersprache und eine Entwicklungsumgebung auf dem Host zum Schreiben und Debuggen der Host-Software. Die Host-Software kann einen Gerätetreiber oder Filtertreiber und/oder Anwendungscode umfassen. Um einen Gerätetreiber schreiben zu können, benötigen Sie das Windows-DDK (Driver Development Kit) von Microsoft.
- Weiterhin empfehlenswert sind ein Monitorprogramm zum Debuggen der Geräte-Firmware und ein Protokollanalysator zur Betrachtung des USB-Datenverkehrs.

1.5.3 Schritte der Entwicklung eines Projekts

Unabhängig von der Größe eines Projekts ist es immer zweckmäßig, es modular zu erstellen und zunächst jeweils einzelne Module zum Laufen zu bringen, bevor man zu den nächsten übergeht. Beim Schreiben der Firmware kann man damit beginnen, nur so viel Code zu schreiben, dass Windows das Gerät erkennen und enumerieren kann. Wenn dies funktioniert, können kleine Datenblöcke mit Anwendungsprogrammen ausgetauscht werden. Anschließend kann man nach und nach anwendungsspezifischen Code hinzufügen. Die Schritte der Projektentwicklung umfassen die Anfangsentscheidungen, die Enumeration und den Datenaustausch:

Anfangsentscheidungen

Bevor Sie mit der eigentlichen Entwicklung beginnen können, müssen Sie Daten sammeln und einige Entscheidungen treffen:

1. Spezifizieren Sie die Anforderungen an das Gerät. Wie viel Daten müssen über die USB-Schnittstelle übertragen werden und in welcher Geschwindigkeit soll dies geschehen? Wird Fehlerkorrektur benötigt? Wie viel Energie wird das Gerät verbrauchen? Was muss das Gerät sonst noch können?
2. Anhand der Anforderungen können Sie entscheiden, ob der PC bei der Kommunikation mit dem Peripheriegerät Windows-eigene USB-Treiber, generische Gerätetreiber anderer Herkunft oder speziell angepasste Treiber verwenden soll.
3. Wählen Sie einen Controllerchip aus, der den Anforderungen entspricht.

Enumeration

Folgendes ist erforderlich, damit Windows Ihr Gerät enumerieren kann:

1. Schreiben Sie den Code, der vom Controllerchip benötigt wird, um vom Host enumeriert zu werden. Die Einzelheiten sind zwar von Chip zu Chip unterschiedlich, doch muss jeder Chip eine Reihe von Deskriptoren zum Host übertragen können. Bei den Deskriptoren handelt es sich um Datenstrukturen, die die USB-Fähigkeiten des Geräts und deren Einsatz beschreiben. Der Chip benötigt zudem Programmcode oder Hardware zur Entschlüsselung und Beantwortung der Anforderungen, die der Host bei der Enumeration des Geräts sendet, und anderer auftretender Ereignisse. Im Allgemeinen stellen die Chipanbieter Beispielcode bereit, den Sie anpassen können. Einige Controller können enumeriert werden, ohne dass dazu Anwendercode benötigt wird.
2. Finden oder erstellen Sie eine INF-Datei, mit der Windows das Gerät erkennen kann und ordnen Sie ihm einen Treiber zu. Die INF-Datei ist eine Textdatei, in der der Treiber angegeben wird, der auf dem Host für das Gerät verwendet werden soll. Wenn das Gerät einer der von Windows unterstützten Klassen entspricht, dann lässt sich möglicherweise eine der zusammen mit Windows gelieferten INF-Dateien benutzen.
3. Falls erforderlich, entwickeln und erstellen Sie Schaltungen zum Testen des Chips und der Firmware. Häufig können Sie dazu vom Chiphersteller angebotene Entwicklungsplatinen verwenden.
4. Laden Sie den Code in das Gerät und schließen Sie es an den Host-Bus an. Nun sollte Windows das Gerät enumerieren, zur Systemsteuerung hinzufügen und es korrekt erkennen.
5. Unterziehen Sie das Gerät der Fehlersuche und wiederholen Sie bei Bedarf die obigen Schritte!

Datenaustausch

Wenn das Gerät erfolgreich enumeriert werden konnte, dann können Sie weitere Komponenten und Code hinzufügen, die dem Gerät die Ausführung seiner geplanten Funktionen ermöglichen. Bei Bedarf schreiben Sie Anwendungscode, um mit dem Gerät zu kommunizieren und es zu testen. Nach dem Debuggen des Codes, können Sie ihn in den Chip einprogrammieren und ihn zusammen mit der endgültigen Hardware testen.

Bevor Sie damit tatsächlich beginnen, sollten Sie allerdings ein wenig mehr darüber wissen, wie der Host Geräte enumeriert und Daten zu ihnen überträgt, um die richtige Entscheidung hinsichtlich Controllerchips und Treiber treffen zu können. Dieser Entscheidung widmen sich die folgenden vier Kapitel.