USB - unbekannter serieller Bus

eine vereinfachte Einführung in die Funktionsweise des Universal Serial Bus

Stefan Schürmans <1stein@schuermans.info>

USB - unbekannter serieller Bus

USB - Universal Serial Bus

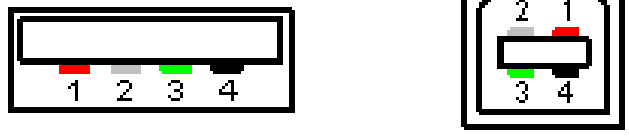
- ≈ 1995 von Intel erfunden
- heute HP, Intel, Microsoft, NEC, Philips, ...
- soll in Zukunft alle anderen Schnittstellen ablösen
  - PS/2-Tastatur, PS/2-Maus
  - RS232, Parallelport (Hacker-Port)

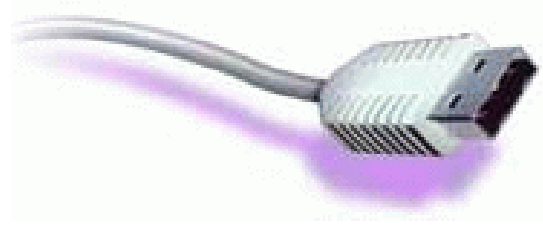
USB - unbekannter serieller Bus

- fast jeder Computer hat es
- fast jeder benutzt es
- fast keiner weiss, wie es funktioniert

USB - Eigenschaften

- einheitliche Stecker für alle Geräte

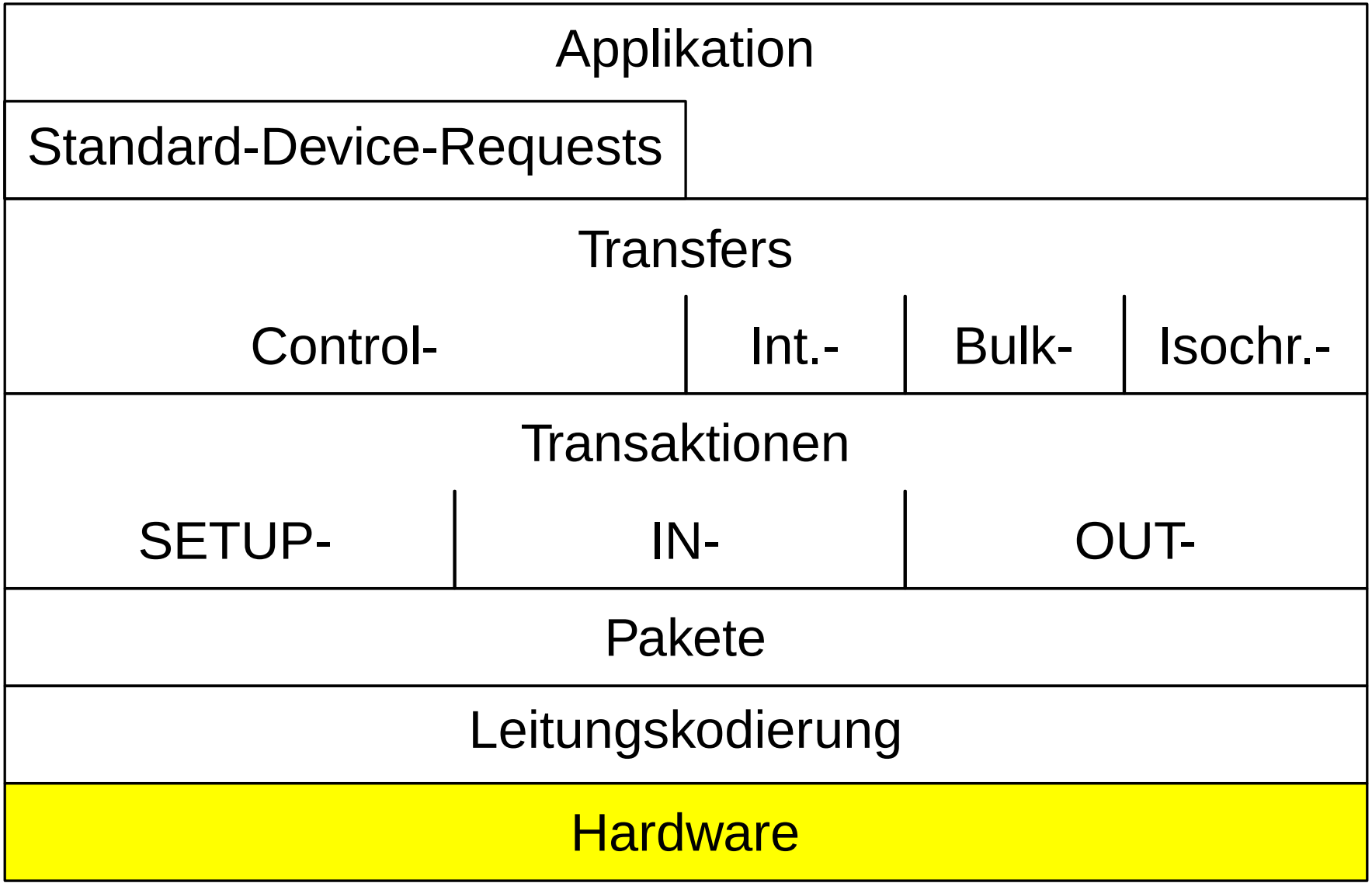




- Stromversorgung der Geräte
- Baum-Verkabelung
  - für weniger Kabelsalat

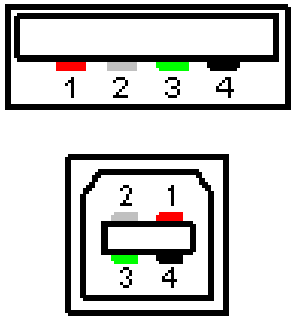
- hot-plugable
- verschiedene Geschwindigkeiten
  - Low-Speed: $1.5Mbps$
  - Full-Speed: $12Mbps$
  - High-Speed: $480Mbps$ (USB 2.0)

USB - Protokoll-Stack 1/7



USB - Pins und Strom

Pinbelegung:



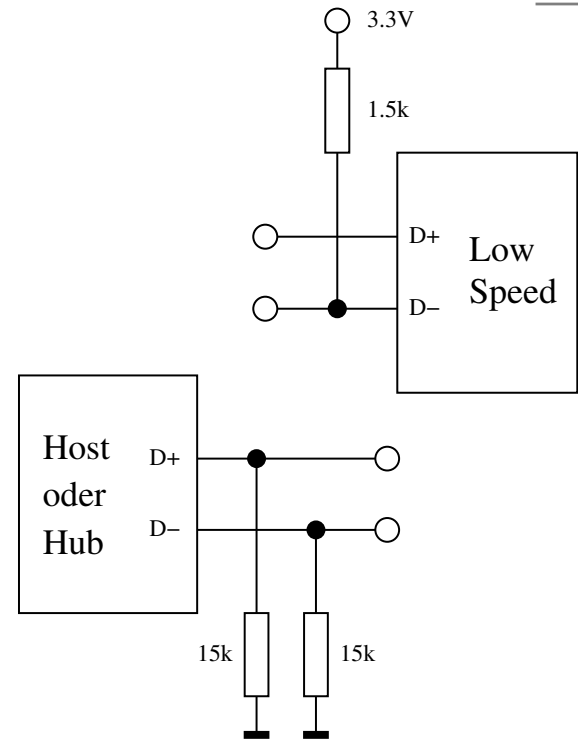
Pin	Farbe	Symbol	Funktion
1 (länger)	rot	V_{BUS}	5V
2	weiss	$D-$	Datenleitung
3	grün	$D+$	Datenleitung
4 (länger)	schwarz	GND	Masse

Stromversorgung:

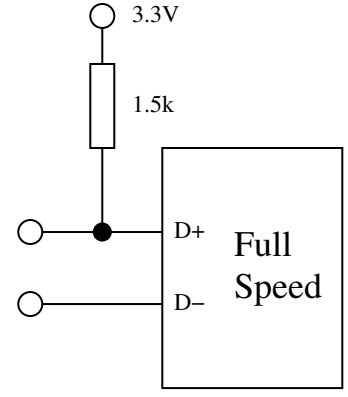
Modus	max. Stromaufnahme
nicht konfiguriert	$100mA$
konfiguriert	bis zu $500mA$
Suspend-Modus	$2.5mA$

USB - Leitungszustände

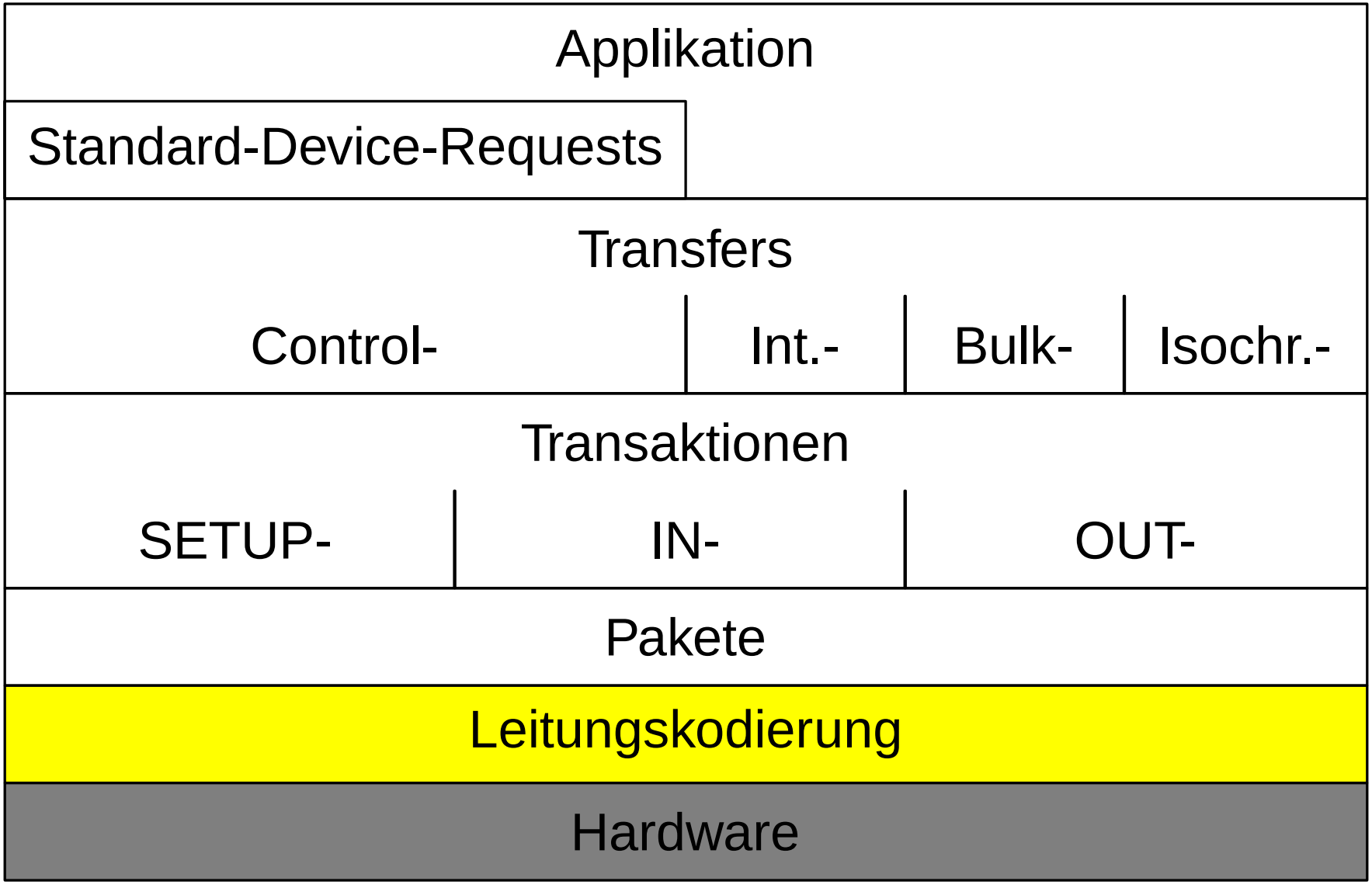
Zustand	Treiber	Low Speed		Full Speed	
		$D+$	$D-$	$D+$	$D-$
$Idle$	passiv	Low	High	High	Low
J	aktiv	Low	High	High	Low
K	aktiv	High	Low	Low	High
$SE0$	aktiv	Low	Low	Low	Low



Ereigniss	Abkürzung	Leitungszustand
Start of Packet	SOP	Übergang $Idle \rightarrow K$
End of Packet	EOP	2 Takte $SE0$, 1 Takt $Idle$
Reset	RST	mehr als $2.5\mu s SE0$



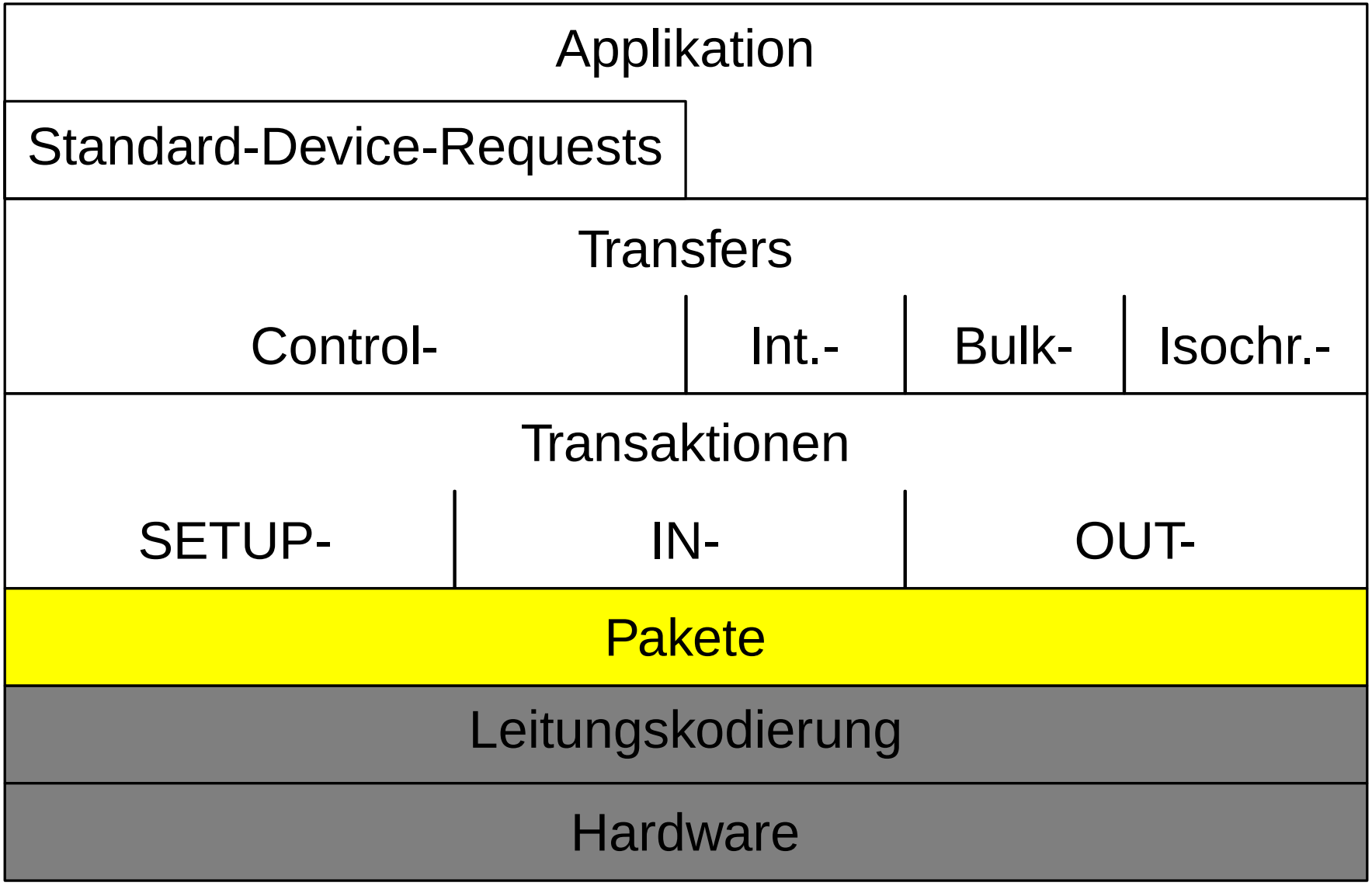
USB - Protokoll-Stack 2/7



USB - Leitungskodierung

Schritt	Funktion	Beispiel
		`0xE1,0x17`
	Bitstrom erzeugen ($LSB\ 1^{st}$)	$1000\,0111\,1110\,1000$
Bit-Stuffer	nach 6 Einsen 1 Null einfügen	$1000\,0111\,1110\,0100\,0$
NRZI-Encoder	Null: Wechsel J/K	$(Idle)\ JK\,JK\,JJ\,JJ$
	Eins: kein Wechsel J/K	$JJ\,JK\,JJ\,KJ\,K$
Übertragung		
NRZI-Decoder	Wechsel J/K: Null	$1000\,0111\,1110\,0100\,0$
	kein Wechsel J/K: Eins	
Bit-Destuffer	nach 6 Einsen 1 Null entfernen	$1000\,0111\,1110\,1000$
	Bitstrom interpretieren	`E1,0x17`

USB - Protokoll-Stack 3/7



USB - Adressen und Endpoints

- USB-Gerät hat eine Adresse (ADDR) von 1 bis 127
  - nach dem Einstecken ist Adresse 0
  - Adresse wird vom Host während Konfiguration gesetzt
- USB-Gerät hat mehrere Kommunikationskanäle, die Endpoints (EP)
  - Endpoint 0 (EP0)
    - für Konfiguration
    - auf jedem Gerät verfügbar
  - weitere Endpoints (EP1 - EP15)
    - je nach Anwendung
- Informationen über USB-Gerät
  - in sogenannten Descriptors im Gerät gespeichert
  - Abfrage durch Host über EP0

USB - Pakete 1/5

Start of Frame

SYNC	SOF	FRAME #	CRC5	EOP
00000001	0xA5	0x7FF	0x02	EOP

- jede Millisekunde zur Synchronisation vom Host gesendet
- verhindert, dass Geräte in Suspend-Modus gehen

Setup

SYNC	SETUP	ADDR	EP	CRC5	EOP
00000001	0x2D	0x01	0x00	0x17	EOP

- Host kündigt das Senden von Setup-Daten an
- z.B. über EP0 benutzt
- es folgt ein DATAx-Paket vom Host

USB - Pakete 2/5

Input

SYNC	IN	ADDR	EP	CRC5	EOP
00000001	0x69	0x03	0x01	0x07	EOP

- Host fordert ein Gerät zum Senden von Daten auf
- es folgt ein DATAx-Paket vom Gerät

Output

SYNC	OUT	ADDR	EP	CRC5	EOP
00000001	0xE1	0x02	0x02	0x01	EOP

- Host kündigt das Senden von Daten an
- es folgt ein DATAx-Paket vom Host

USB - Pakete 3/5

Daten

SYNC	DATA0	DATA	CRC16	EOP
00000001	0xC3	0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77	0xCBA8	EOP

SYNC	DATA1	DATA	CRC16	EOP
00000001	0x4B	0x88 0x99 0xAA 0xBB 0xCC 0xDD 0xEE 0xFF	0x8705	EOP

- Daten von einem Gerät oder vom Host
  - vom Host nach SETUP- oder OUT-Paket gesendet
  - vom Gerät nach empfangenem IN-Paket gesendet
- Anzahl der Datenbytes:
  - $0..8$ bei Low-Speed
  - $0..64$ bei Full-Speed
- DATA0 und DATA1 immer abwechselnd gesendet (Synchronisation)

USB - Pakete 4/5

Acknowledge

SYNC	ACK	EOP
00000001	0xD2	EOP

- Bestätigung des Empfangs eines DATAx-Pakets
- vom Host oder von einem Gerät gesendet

Not-Acknowledge

SYNC	NAK	EOP
00000001	0x5A	EOP

- nur von Geräten gesendet
  - DATAx empfangen und Daten können nicht verarbeitet werden
  - IN empfangen und keine Daten vorhanden

USB - Pakete 5/5

Stall

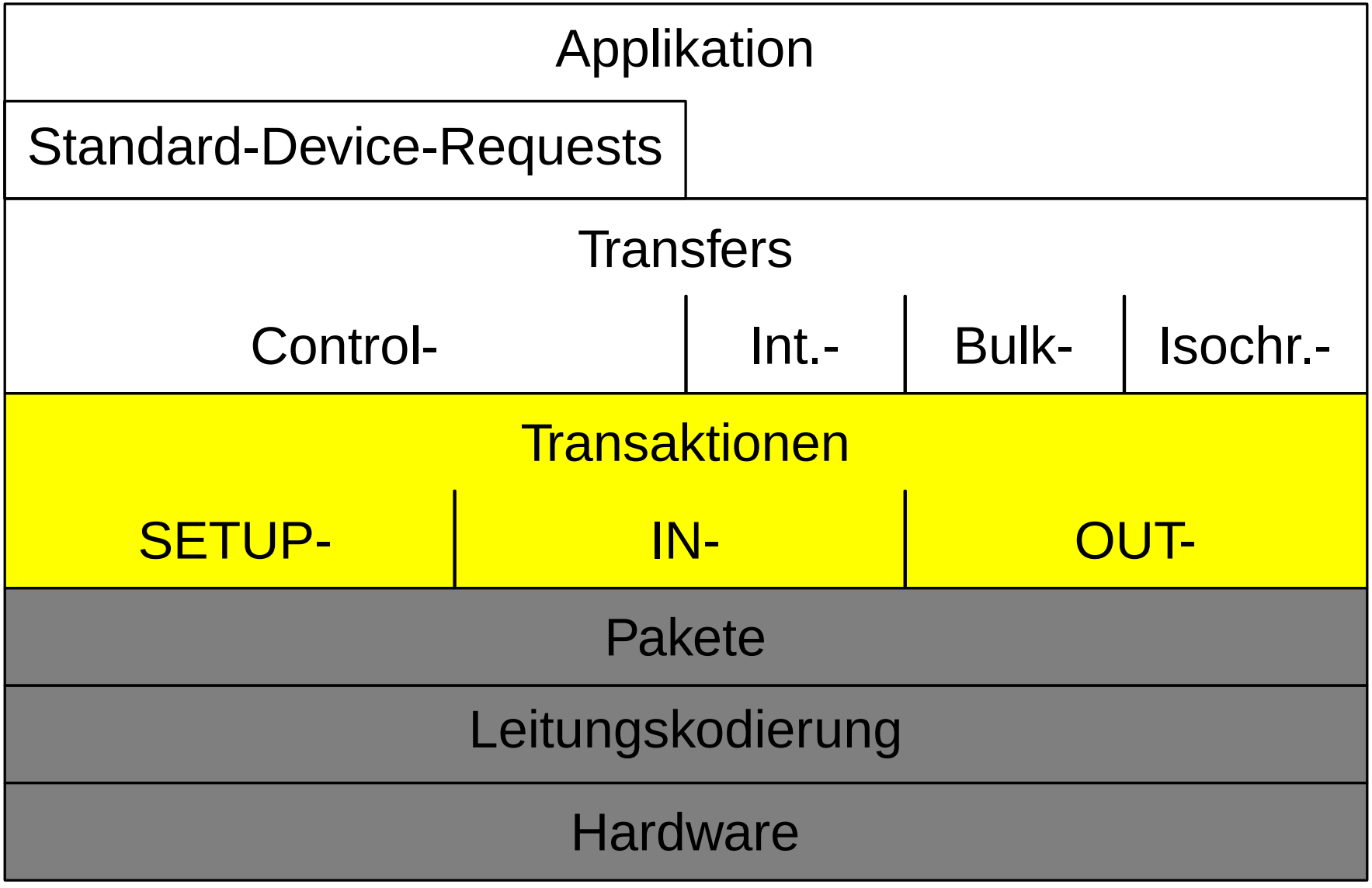
SYNC	STALL	EOP
00000001	0x1E	EOP

- nur von Geräten gesendet
  - DATAx empfangen und Daten können nicht verarbeitet werden
  - IN empfangen und keine Daten vorhanden
- Aktion des Hosts erforderlich um Situation zu ändern

weitere Pakete

- spezielle Pakete für Low-Speed
  - Low-Speed Start of Frame (EOP)
  - Low-Speed Tunnelig over Full-Speed (PRE)
- . . .

USB - Protokoll-Stack 4/7



USB - SETUP-Transaktion

Übertragung eines Setup-Daten-Pakets an ein Gerät

SYNC	SETUP	ADDR	EP	CRC5	EOP
00000001	0x2D	0x02	0x00	0x15	EOP

SYNC	DATA0	DATA	CRC16	EOP
00000001	0xC3	0x80 0x06 0x00 0x01 0x00 0x00 0x12 0x00	0xD768	EOP

SYNC	ACK	EOP
00000001	0xD2	EOP

- benutzt zur Konfiguration der Geräte
- z.B. auf EP0

Konvention:

- grün: vom Host gesendete Pakete
- rot: von einem Gerät gesendete Pakete

USB - IN-Transaktion 1/2

Übertragung eines Daten-Pakets von einem Gerät an den Host

SYNC	IN	ADDR	EP	CRC5	EOP
00000001	0x69	0x02	0x02	0x01	EOP

SYNC	DATA0	DATA	CRC16	EOP
00000001	0xC3	0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77	0xCBA8	EOP

SYNC	ACK	EOP
00000001	0xD2	EOP

- Gerät ist bereit und antwortet mit DATA0
- nächstes Paket wird mit DATA1 gesendet

USB - IN-Transaktion 2/2

Übertragung eines Daten-Pakets von einem Gerät an den Host

SYNC	IN	ADDR	EP	CRC5	EOP
00000001	0x69	0x02	0x02	0x01	EOP

SYNC	NAK	EOP
00000001	0x5A	EOP

- Gerät ist (noch) nicht bereit und antwortet mit NAK
- Host sendet im nächsten Frame ($1ms$) erneut ein IN

USB - OUT-Transaktion 1/2

Übertragung eines Daten-Pakets vom Host an ein Gerät

SYNC	OUT	ADDR	EP	CRC5	EOP
00000001	0xE1	0x02	0x02	0x01	EOP

SYNC	DATA0	DATA	CRC16	EOP
00000001	0xC3	0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77	0xCBA8	EOP

SYNC	ACK	EOP
00000001	0xD2	EOP

- Gerät ist bereit und antwortet mit ACK
- nächstes Paket wird mit DATA1 gesendet

USB - OUT-Transaktion 2/2

Übertragung eines Daten-Pakets vom Host an ein Gerät

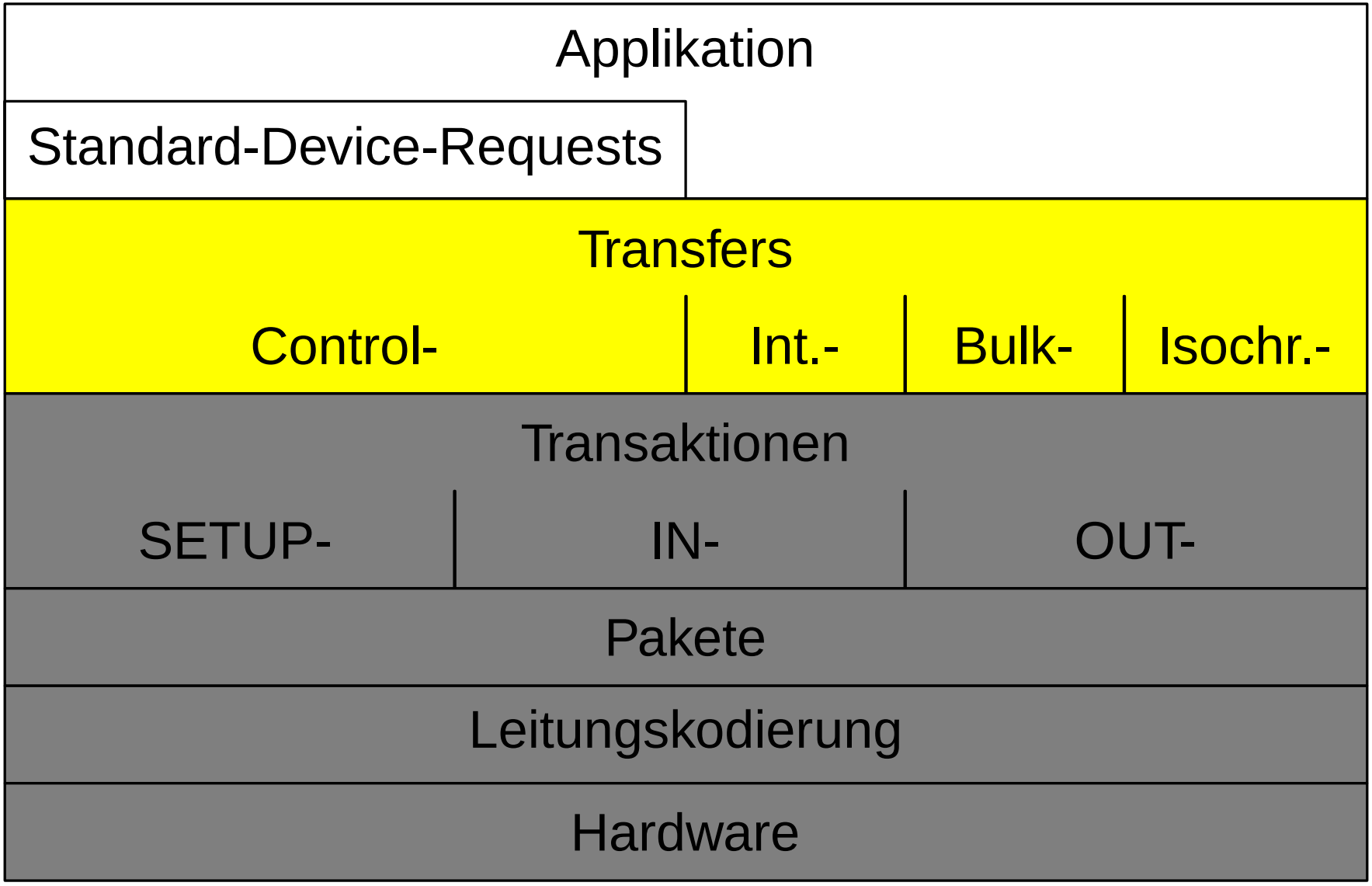
SYNC	OUT	ADDR	EP	CRC5	EOP
00000001	0xE1	0x02	0x02	0x01	EOP

SYNC	DATA0	DATA	CRC16	EOP
00000001	0xC3	0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77	0xCBA8	EOP

SYNC	NAK	EOP
00000001	0x5A	EOP

- Gerät ist (noch) nicht bereit und antwortet mit NAK
- Host sendet im nächsten Frame ($1ms$) erneut ein DATA0 mit den gleichen Daten

USB - Protokoll-Stack 5/7



USB - Control-Transfer

- Konfiguration eines Geräts
  - auf EP0, bidirektional
  - insgesamt bis zu 10% der verfügbaren Bandbreite
  - Wiederholung bei Übertragungsfehler
- Set...: Daten vom Host zum Gerät
  - Setup-Stage: SETUP-Transaktion (DATA0)
  - Data-Stage: $0..n$ OUT-Transaktionen (DATA1, DATA0, ...)
  - Status-Stage: 0-Byte IN-Transaktion (DATA1)
- Get...: Daten vom Gerät zum Host
  - Setup-Stage: SETUP-Transaktion (DATA0)
  - Data-Stage: $1..n$ IN-Transaktionen (DATA1, DATA0, ...)
  - Status-Stage: 0-Byte OUT-Transaktion (DATA1)

USB - Interrupt-Transfer

- Periodischer Datenaustausch, kein echter Interrupt
  - auf EP1 - EP15, unidirektional
  - eine Transaktion pro Zeittakt
    - $1ms$, $2ms$, $4ms$, ..., $128ms$
  - insgesamt bis zu 90% der verfügbaren Bandbreite
  - Wiederholung bei Übertragungsfehler
- Interrupt-Out: Daten vom Host zum Gerät
  - eine OUT-Transaktion pro gewähltem Zeittakt
- Interrupt-In: Daten vom Gerät zum Host
  - eine IN-Transaktion pro gewähltem Zeittakt
- Verwendung
  - Statusabfrage
  - Übertragung kleiner Datenmengen

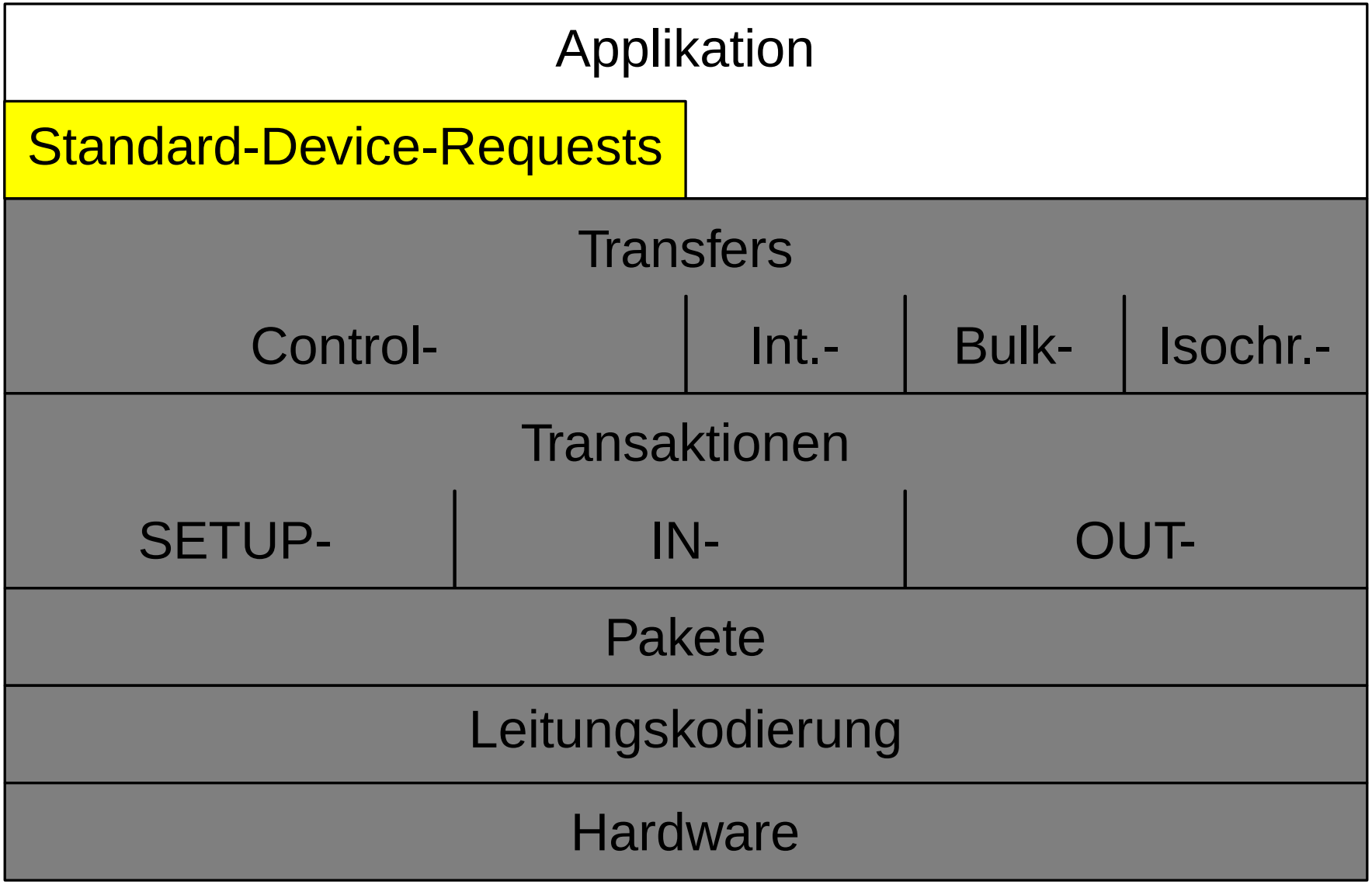
USB - Bulk-Transfer

- zeitunkritischer Austausch großer Datenmengen
  - auf EP1 - EP15, unidirektional
  - beliebig viele Transaktion zu jedem Zeitpunkt
  - kann gesamte freie Bandbreite nutzen
  - Wiederholung bei Übertragungsfehler
  - bei Low-Speed nicht verfügbar
- Bulk-Out: Daten vom Host zum Gerät
  - beliebig viele OUT-Transaktion zu jedem Zeitpunkt
- Bulk-In: Daten vom Gerät zum Host
  - beliebig viele IN-Transaktion zu jedem Zeitpunkt
- Verwendung
  - Datenübertragung
  - Datenspeicher

USB - Isochronous-Transfer

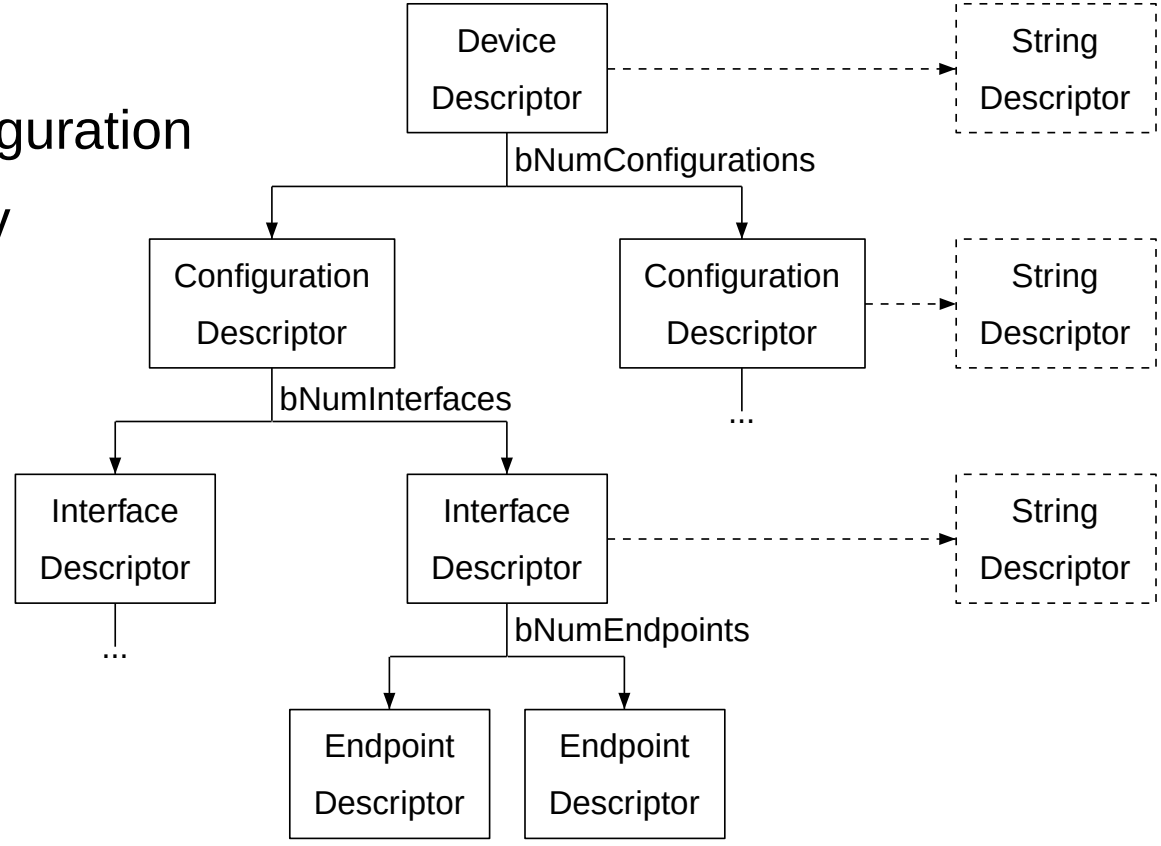
- zeitkritischer Datenaustausch mit konstanter Bandbreite
  - auf EP1 - EP15, unidirektional
  - immer gleiche viele Transaktionen pro Frame ($1ms$)
  - insgesamt bis zu 90% der verfügbaren Bandbreite
  - Übertragungsfehler führt zu Datenverlust
  - bei Low-Speed nicht verfügbar
- Isochronous-Out: Daten vom Host zum Gerät
  - n OUT-Transaktion pro Frame, ohne ACK/NAK
- Isochronous-In: Daten vom Gerät zum Host
  - n IN-Transaktion pro Frame, ohne ACK/NAK
- Verwendung
  - Audio-Daten
  - Video-Daten

USB - Protokoll-Stack 6/7



USB - Descriptors

- Device Descriptor
  - Identität des Geräts
- Configuration Descriptor
  - eine mögliche Konfiguration
  - immer nur eine aktiv
- Interface Descriptor
  - eine Schnittstelle
  - mehrere können aktiv sein
- String Descriptor
  - menschenlesbare Zusatzinformation



USB - Device Descriptor

Offset	Bezeichnung	Länge	Beschreibung	z.B.
0	bLength	1	Länge des Descriptors	0x12
1	bDescriptorType	1	Typ = Device Descriptor	0x01
2	bcdUSB	2	USB-Version 1.1	0x0110
4	bDeviceClass	1	Geräte-Klassen-ID	0x00
5	bDeviceSubClass	1	Geräte-Unterklassen-ID	0x00
6	bDeviceProtocol	1	Geräte-Protokoll-ID	0x00
7	bMaxPacketSize0	1	max. Paketlänge auf EP0	0x08
8	idVendor	2	Hersteller-ID	0x6666
10	idProduct	2	Produkt-ID	0x2342
12	bcdDevice	2	Produkt-Version	0x0100
14	iManufacturer	1	String-Nr. Hersteller-Name	0x01
15	iProduct	1	String-Nr. Produkt-Name	0x02
16	iSerialNumber	1	String-Nr. Seriennummer	0x00
17	bNumConfigurations	1	Anzahl Konfigurationen	0x01

USB - Configuration Descriptor

Offset	Bezeichnung	Länge	Beschreibung	z.B.
0	bLength	1	Länge des Descriptors	0x09
1	bDescriptorType	1	Typ = Configuration Descriptor	0x02
2	wTotalLength	2	Gesamtlänge der Konfiguration	0x19
4	bNumInterfaces	1	Anzahl der Interfaces	0x01
5	bConfigurationValue	1	Nr. dieser Konfiguration	0x01
6	iConfiguration	1	String-Nr. dieser Konfiguration	0x00
7	bmAttributes	1	Bit 7: Stromversorgung über USB	0x80
			Bit 6: eigene Stromversorgung	
			Bit 5: unterstützt Remote-Wakeup	
8	MaxPower	1	Stromaufnahme in $2mA$-Einheiten	0xC8

- zu einer Konfiguration gehören die Interface Descriptors und Endpoint Descriptors dazu

USB - Interface Descriptor

Offset	Bezeichnung	Länge	Beschreibung	z.B.
0	bLength	1	Länge des Descriptors	0x09
1	bDescriptorType	1	Typ = Interface Descriptor	0x04
2	bInterfaceNumber	1	Nr. des Interfaces	0x00
3	bAlternateSetting	1	Nr. der Einstellung	0x00
4	bNumEndpoints	1	Anzahl Endpoints	0x01
5	bInterfaceClass	1	Interface-Klassen-ID	0x00
6	bInterfaceSubClass	1	Interface-Unterklassen-ID	0x00
7	bInterfaceProtocol	1	Interface-Protokoll-ID	0x00
8	iInterface	1	String-Nr. Interface-Name	0x00

- ein Interface kann mehrere Einstellungen besitzen
  - siehe bAlternateSetting
  - z.B. Audio-Übertragung mit unterschiedlicher Datenrate

USB - Endpoint Descriptor

Offset	Bezeichnung	Länge	Beschreibung	z.B.
0	bLength	1	Länge des Descriptors	0x07
1	bDescriptorType	1	Typ = Endpoint Descriptor	0x05
2	bEndpointAddress	1	Endpoint-Nummer	0x81
		1	Bit 7: 0=Output, 1=Input	
3	bmAttributes	1	Transferart	0x03
			0x00: Control-Transfer	
			0x01: Isochronous-Transfer	
			0x02: Bulk-Transfer	
			0x03: Interrupt-Transfer	
4	wMaxPacketSize	2	maximale Paketlänge	0x08
6	bIntervall	1	Abfrage-Interval (für Interrupt-Transfer)	0x10

USB - String Descriptor

String Descriptor 0:

Offset	Bezeichnung	Länge	Beschreibung	z.B.
0	bLength	1	Länge des Descriptors	0x04
1	bDescriptorType	1	Typ = String Descriptor	0x03
2	wUnicode	2	UNICODE-Sprachindex	0x0409 (Englisch)

String Descriptor $1..255$:

Offset	Bezeichnung	Länge	Beschreibung	z.B.
0	bLength	1	Länge des Descriptors	0x08
1	bDescriptorType	1	Typ = String Descriptor	0x03
2	wUnicode	2	UNICODE-Zeichen	0x0066 (“f”)
4	wUnicode	2	UNICODE-Zeichen	0x006F (“o”)
6	wUnicode	2	UNICODE-Zeichen	0x006F (“o”)

USB - Standard-Device-Requests

- Control-Transfers auf EP0 zur Konfiguration des Geräts
  - setzen der Adresse
  - Abfrage der Eigenschaften
  - Einstellen der aktuellen Konfiguration
-

Offset	Bezeichnung	Länge	Beschreibung
0	bmRequestType	1	Bit 7: 0=Output, 1=Input
			Bit 6,5: 0=Standard-Request
			Bits 4..0: 0=Device, 1=Interface, 2=Endpoint
1	bRequest	1	Kenn-Nr. des Requests
2	wValue	2	(unterschiedlich)
4	wIndex	2	(unterschiedlich)
6	wLength	2	(Maximal-)Länge der Daten

- hier nur Überblick über die wichtigsten
- zusätzlich benutzerdefinierte Device-Requests

USB - GetStatus

bmRequestType	bRequest	wValue	wIndex	wLength
1000 0000 b	0x00	0x0000	0x0000	0x0002
1000 0001 b	0x00	0x0000	Interface-Nr.	0x0002
1000 0010 b	0x00	0x0000	Endpoint-Nr.	0x0002

- lese Status des Geräts / eines Interfaces / eines Endpoints
- Gerät antwortet mit Statusinformation:
  -

	Bit	Bedeutung
Device	0	1 = SelfPower
	1	1 = RemoteWakeup
	2..15	(immer 0)
Interface	0..15	(immer 0)
Endpoint	0	1 = Stall
	1..15	(immer 0)

USB - Clear/SetFeature

ClearFeature:

bmRequestType	bRequest	wValue	wIndex	wLength
0000 0000 b	0x01	Feature	0x0000	0x0000
0000 0001 b	0x01	Feature	Interface-Nr.	0x0000
0000 0010 b	0x01	Feature	Endpoint-Nr.	0x0000

SetFeature:

bmRequestType	bRequest	wValue	wIndex	wLength
0000 0000 b	0x03	Feature	0x0000	0x0000
0000 0001 b	0x03	Feature	Interface-Nr.	0x0000
0000 0010 b	0x03	Feature	Endpoint-Nr.	0x0000

- schalte Eigenschaft des Geräts / Interfaces / Endpoints aus bzw. ein
- keine weitere Daten werden übertragen
-

	Feature	Bedeutung
Endpoint	0x0000	Stall
Device	0x0001	RemoteWakeup zulässig

USB - SetAddress

bmRequestType	bRequest	wValue	wIndex	wLength
0000 0000 b	0x05	neue Adresse	0x0000	0x0000

- setze Adresse des Geräts
- keine weitere Daten werden übertragen
- gesamter Control-Transfer läuft noch mit alter Adresse
- nach dem Einstecken hat Gerät immer Adresse 0

USB - GetDescriptor

bmRequestType	bRequest	wValue	wIndex	wLength
1000 0000 b	0x06	DescrTyp : DescrIndex	0x0000	Maximallänge

- lese einen Descriptor
DescrTyp	
0x01	Device-Descriptor
0x02	Configuration-Descriptor (mit Unterdescriptoren)
0x03	String-Descriptor mit Nr. DescrIndex
- Gerät antwortet mit Daten des Descriptors
  - jedoch maximal mit den ersten wLength Bytes

USB - Get/SetConfiguration

GetConfiguration:

bmRequestType	bRequest	wValue	wIndex	wLength
1000 0000 b	0x08	0x0000	0x0000	0x0001

- lese aktuelle Konfiguration
- Gerät antwortet mit der Nummer der aktiven Konfiguration
  - $0x00$ falls nicht konfiguriert
  - bConfigurationValue falls konfiguriert

SetConfiguration:

bmRequestType	bRequest	wValue	wIndex	wLength
0000 0000 b	0x09	bConfigurationValue	0x0000	0x0000

- setze neue Konfiguration
- keine weitere Daten werden übertragen
- nur EP0 darf aktiv sein, falls Gerät nicht konfiguriert ist

USB - Get/SetInterface

GetInterface:

bmRequestType	bRequest	wValue	wIndex	wLength
1000 0001 b	0x0A	0x0000	Interface-Nr.	0x0001

- lese aktuelles AlternateSetting eines Interfaces
- Gerät antwortet mit der Nummer des aktuellen AlternateSetting

SetInterface:

bmRequestType	bRequest	wValue	wIndex	wLength
0000 0001 b	0x0B	AlternateSetting	Interface-Nr.	0x0000

- setze neues AlternateSetting eines Interfaces
- keine weitere Daten werden übertragen

USB - typische Szenarios

Linux:

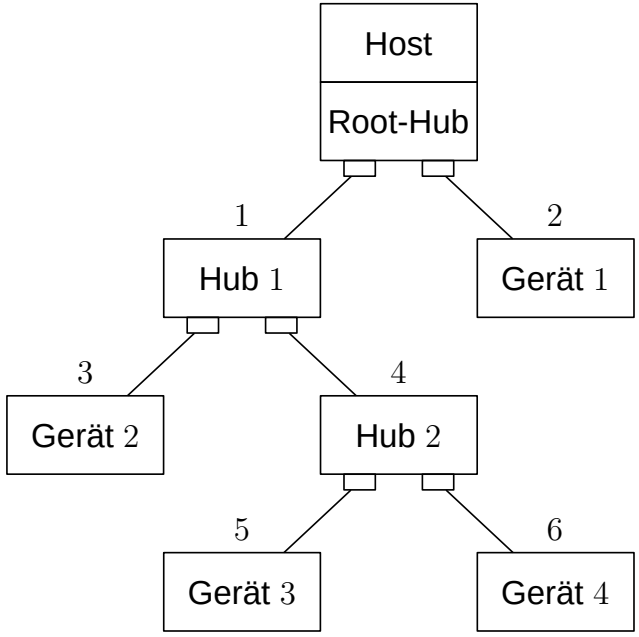
- SetAddress
- GetDescriptor (DeviceD.)
  - wLength=8
- GetDescriptor (DeviceD.)
- GetDescriptor (ConfigD.)
  - evtl. mehrfach
- GetDescriptor (StringD.)
  - entfällt / mehrfach
  - je nach Treiber
- SetConfiguration

Windows:

- GetDescriptor (DeviceD.)
  - wLength=8
- Reset
- SetAddress
- GetDescriptor (DeviceD.)
- GetDescriptor (ConfigD.)
  - evtl. mehrfach
- GetDescriptor (StringD.)
  - meist mehrfach
- SetConfiguration

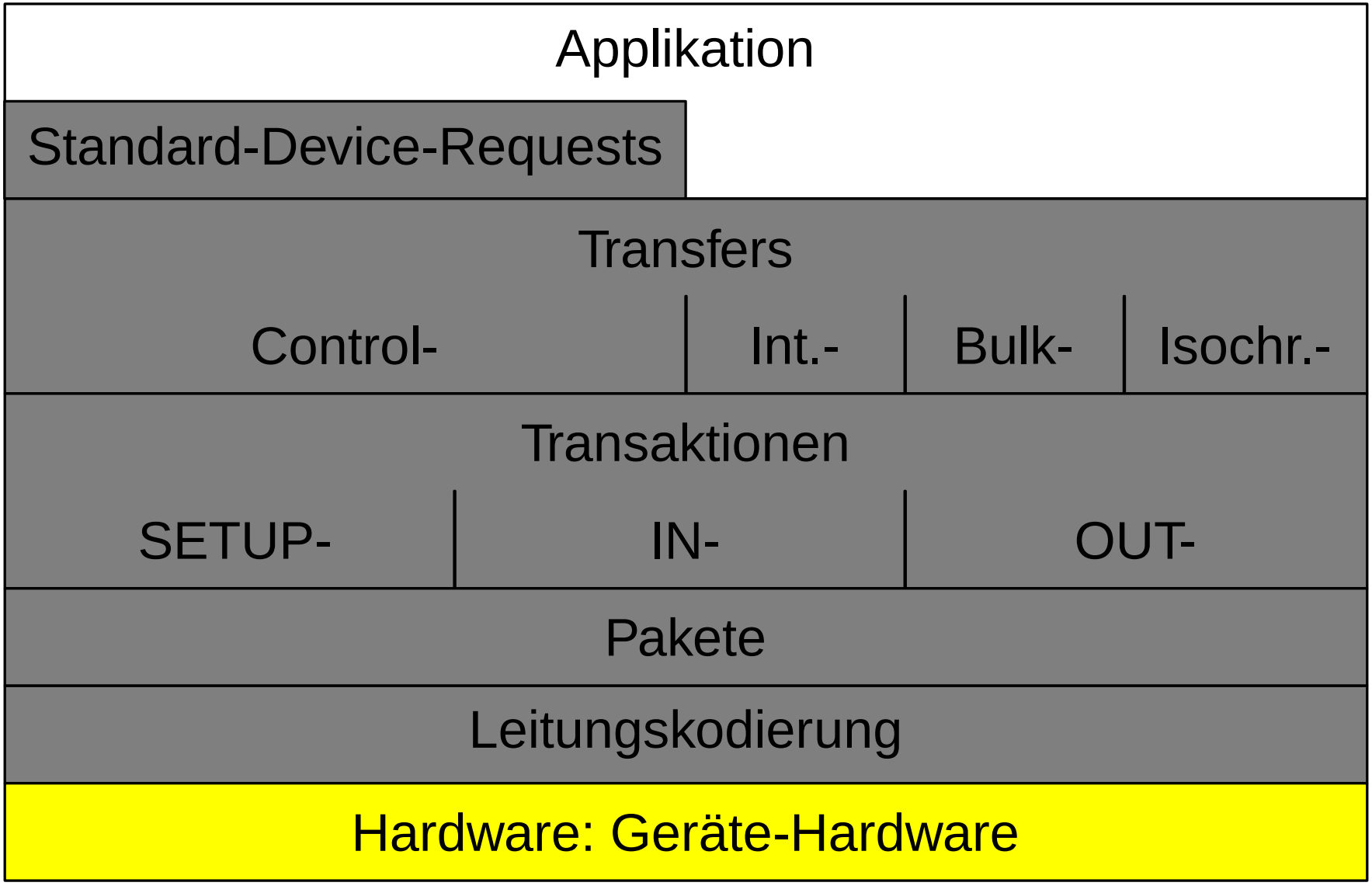
USB - PowerOn-Enumeration

- alle Hub-Ports deaktiviert
- alle Geräte haben Adresse 0



- Enumeration
  - Port 1 des Root-Hub wird aktiviert
  - Hub 1 bekommt Adresse 1
  - Port 2 des Root-Hub wird aktiviert
  - Gerät 1 bekommt Adresse 2
  - Port 1 von Hub 1 wird aktiviert
  - Gerät 2 bekommt Adresse 3
  - Port 2 von Hub 1 wird aktiviert
  - Hub 2 bekommt Adresse 4
  - Port 1 von Hub 2 wird aktiviert
  - Gerät 3 bekommt Adresse 5
  - Port 2 von Hub 2 wird aktiviert
  - Gerät 4 bekommt Adresse 6

USB - Protokoll-Stack 7/7



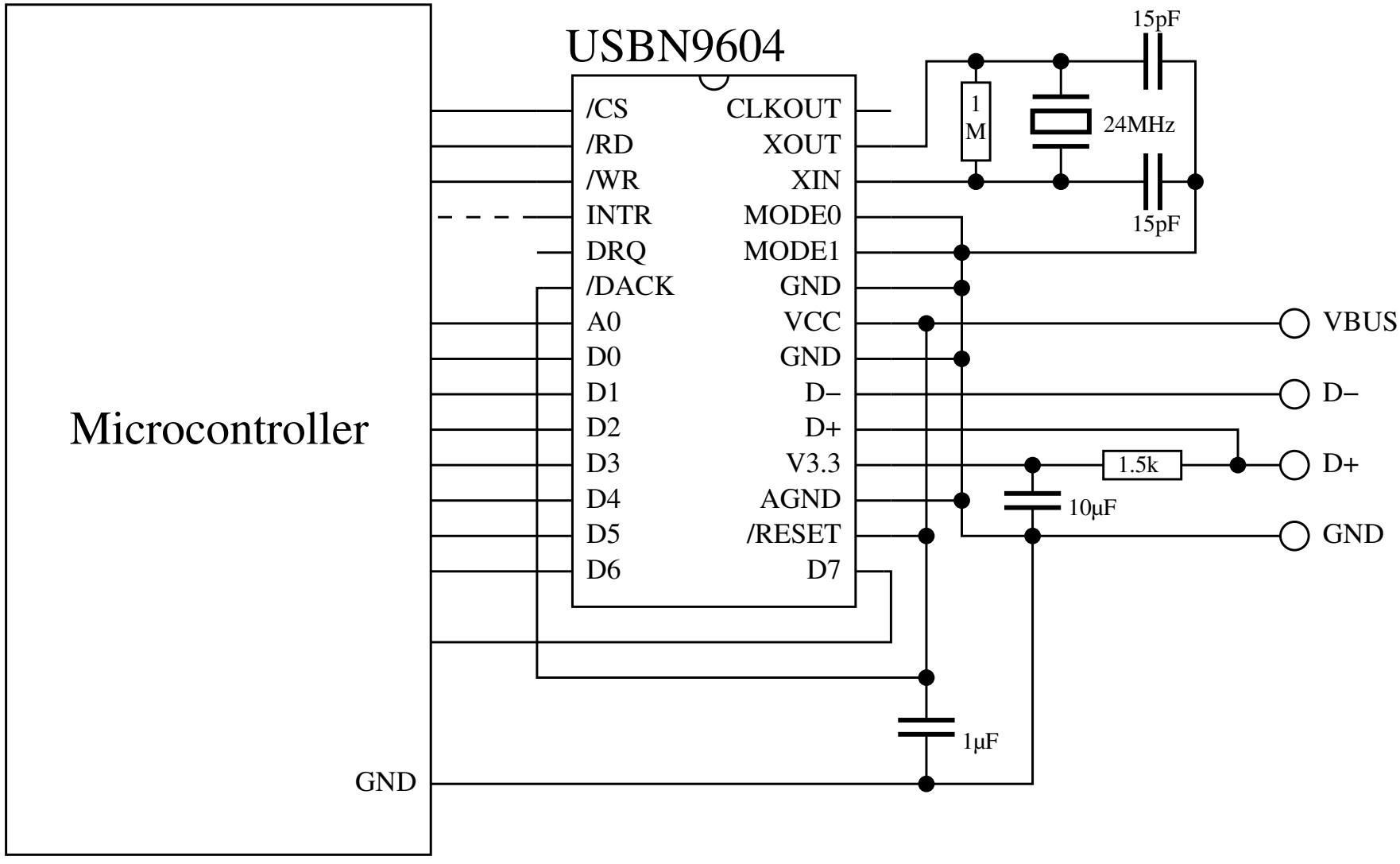
USB - Geräte-Hardware

- USB viel zu schnell für Software-Implementierung
  - zumindest ab Full-Speed ($12Mbps$)
- USB-Interface
  - implementieren USB-Protokoll in Hardware
  - stellen Daten der Endpoints in FIFOs bereit
  - sorgen für Verbindung zwischen Microcontroller und USB
  - einige Full-Speed Chips sogar noch von Hand lötbar
  - z.B. National Semiconductor USBN9604
- USB-Controller
  - USB-Interface in Microcontroller eingebaut
  - ab Full-Speed meist sehr viele und zu kleine Pins
  - z.B. Cypress EZ-USB-Serie

USB - Usb-Interface USBN9604 1/4

- USB 1.1 Full-Speed ($12Mbps$)
  - alle Transferarten
- schnelles paralleles Microcontroller-Interface
  - 8 Bit Adress-/Datenbus
  - Register-Zugriffszeit < $1\mu s$
- integrierter $3.3V$ Regler
  - nur wenige externe Bauteile nötig
- SO-28 Gehäuse
  - $1.27mm$ Pinabstand
  - von Hand lötbar
- ca. 5 Euro bei 1 Stück

USB - Usb-Interface USBN9604 2/4



USB - Usb-Interface USBN9604 3/4

- paralleles Microcontroller-Interface
  - D7..0: Adress-/Datenbus
  - A0: 1=Adresse, 0=Daten
  - /CS: 0=Chip aktiviert
  - /RD: 0=Lese Adresse/Daten
  - /WR: 0=Schreibe Adr./Daten
  - INTR: 1=Interrupt
- Zugriff auf die 64 Register
  - z.B. Chip-Konfiguration
  - z.B. USB-Adresse
  - z.B. Endpoint-Konfiguration
  - z.B. Endpoint-Daten
- Register lesen
  - /CS=1, /RD=1, /WR=1
  - D7..0:=Reg.-Nr., A0:=1
  - /CS:=0, /WR:=0
  - /CS:=1, /WR:=1
  - D7..0:=[input], A0:=0
  - /CS:=0, /RD:=0
  - Daten:=D7..0
  - /CS:=1, /RD:=1

USB - Usb-Interface USBN9604 4/4

Pufferung der Daten vom/zum USB in FIFOs

- 1 bidirektionaler FIFO (8 Byte) für Endpoint 0
  - für Control-Transfers
  - benutzt zur Konfiguration des Geräts
  - immer aktiv
- 3 Output-FIFOs (64 Byte) für 3 Endpoints (EP1 - EP15)
  - für Interrupt-Out, Bulk-Out, Isochronous-Out
  - (de)aktivierbar durch Microcontroller
- 3 Input-FIFOs (64 Byte) für 3 Endpoints (EP1 - EP15)
  - für Interrupt-In, Bulk-In, Isochronous-In
  - (de)aktivierbar durch Microcontroller

USB - weitere Informationen

- http[s]://www.usb.org/
  - u.a. USB-Spezifikation
- USBN9604 Datenblatt
- /usr/src/linux/drivers/usb/usb-skeleton.c
- http[s]://1stein.schuermans.info/hackerport/
  - Beispiel eines selbstgebauten USB-Geräts

Noch Fragen?