Kurze Einführung zu USB

Die Kommunikation über USB soll möglichst einfach dargestellt werden. Die Bibliothek für ATMEL-USB-AVRs verwendet keine Standardklassen. Sie arbeitet auf der PC-Seite mit der freien „libusb“. Die Software und weitere Erklärungen findet man unter www.weigu.lu/usb.

USB-Transfer

Die USB-Übertragung zwischen PC (*host*) und Gerät (*device, function*) besteht aus mehreren Protokollschichten. Die gesamte Übertragung (Kommunikationsanforderung herstellen und ausführen) wird als **USB-Transfer** bezeichnet. Es existieren **vier** unterschiedliche **Transfertypen**[1]:

- **Control**-Transfer:
  zur Identifikation und Steuerung des Gerätes und für herstellerspezifische Anforderungen (muss von jedem USB-Gerät unterstützt werden und arbeitet immer mit Endpunkt 0 (siehe weiter unten))
- **Bulk**-Transfer:
  zur Übertragung großer Datenmengen ohne garantierte Geschwindigkeit (Bsp.: Laufwerk)
- **Interrupt**-Transfer:
  mit garantierter Bandbreite für geringes Datenvolumen (Bsp.: Tastatur)
- **Isochron**-Transfer:
  mit garantierter Bandbreite aber ohne Fehlerkorrektur (Bsp.: Streaming von Audiodaten)

Ein Transfer besteht aus einer oder mehreren **Transaktionen** (Übergabe eines Dienstes an einen Endpunkt). Es gibt drei Arten von Transaktionen[2]:

- **SETUP (Control)**-Transaktion: bidirektionale Nachrichten
- **IN**-Transaktion: Daten vom Gerät zum PC
- **OUT**-Transaktion: Daten vom PC zum Gerät

Jede Transaktion muss ohne Unterbrechung abgeschlossen werden. Jede Transaktion besteht aus drei Paketen (Phasen)[3]:

- **TOKEN**-Paket:
  gibt an um welchen Transaktionstyp es sich handelt (SETUP, IN, OUT)

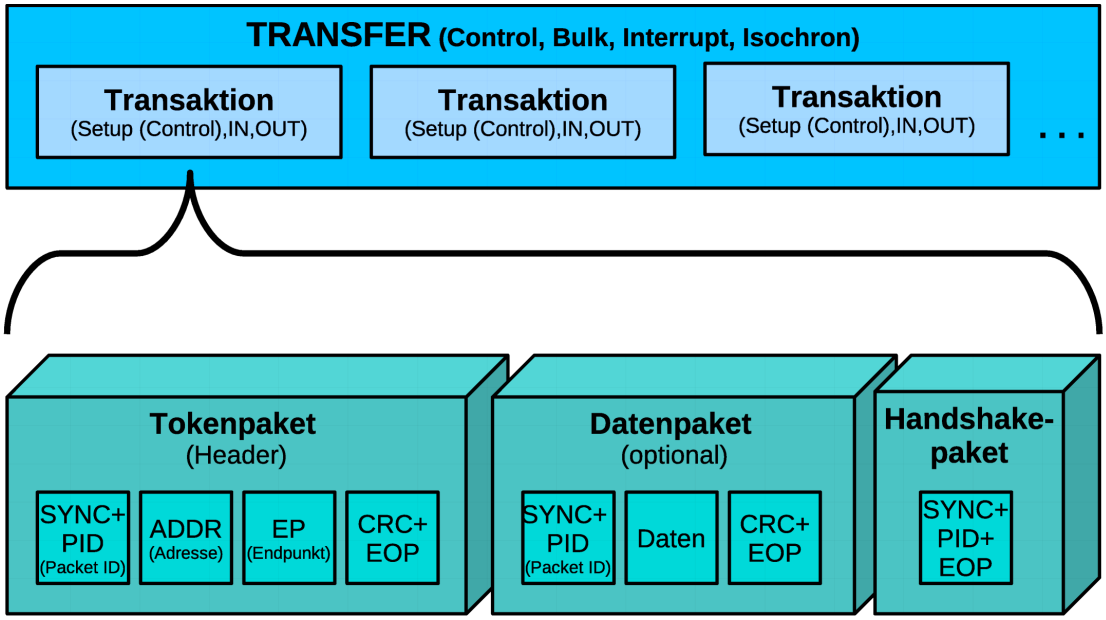
1 Control-Transfers nutzen bidirektionale Nachrichten-Kanäle (Pipes). Die anderen Transfers nutzen unidirektionale Datenstrom-Kanäle.

2 Zur Zeitsteuerung wird auch ein Start-Of-Frame-Transfer bzw. Transaktion durchgeführt. Ebenfalls existiert also ein SOF-Token. Es überträgt eine Zeitreferent (1ms oder 125μs).

3 Es existiert auch eine vierte "Special"-Phase. Auf sie soll hier nicht eingegangen werden.

- (optionale) **DATA**-Paket:
  enthält Daten oder Statusinformationen (DATA0-2, MDATA)
  Speziell: Beim erfolgreichen Abschließen eines Control-Transfers wird als Status-Information (siehe Handshake) ein Datenpaket ohne Daten (**ZLP**, **Z**ero **L**ength data **P**acket) anstelle einer Bestätigung (ACK) versendet.
- **STATUS** (Handshake)-Paket:
  Feedback zur Kommunikation: "Erfolg", "bin beschäftigt (warten)" und "nicht unterstützt" (ACK, NAK, STALL (NYET)).

Die Paketzusammenstellung wird von der Hardware bewerkstelligt. Auf sie soll nicht weiter eingegangen werden.



Endpunkte

Der Datenaustausch passiert zwischen dem PC und einem spezifischen Geräteendpunkt (*endpoint* EP). Ein Endpunkt ist ein Datenspeicher (FIFO[4], Puffer, Speicherbank) im Gerät, der meist nur aus 8-256 Bytes besteht. Jedes Gerät hat mehrere Endpunkte die über die Endpunktadresse (015) angesprochen werden. Diese Adresse enthält in Bit 7 ebenfalls die Richtung der Datenkommunikation[5]. Der Control-Endpunkt 0 ist in jedem Gerät vorhanden und wird für zum Beispiel für die Enumeration benötigt. Er ist als einziger Endpunkt bidirektional. Bei USB 1.1 hat er einen 8 Byte großen FIFO-Speicher. Die Anzahl und mögliche Größe der anderen Endpunkte variiert. Jede Transaktion ist an eine Geräte und eine Endpunktadresse gebunden.

4 FIFO (*First In First Out*): Speicherbank, welche zuerst abgelegte Daten beim Lesen auch als erstes wieder ausgibt, im Gegensatz zum LIFO (*Last In First Out*), der zum Beispiel beim AVR Stapel verwendet wird.

5 Außer dem Control-Endpunkt, da dieser bidirektional; Bit 7 = 0: OUT-Endpunkt; Bit 7 = 1: IN-Endpunkt

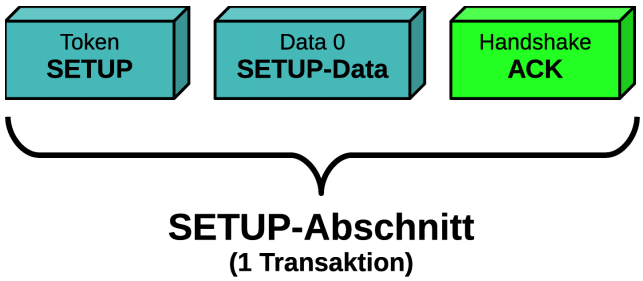
Control Transfer

Die gesamte Enumeration der Gerätes erfolgt mittels Control-Transfers (Standard Anfragen). Control-Transfers können auch für herstellerspezifische Anforderungen genutzt werden. Der Control-Transfer besteht aus drei Abschnitten (*stages*):

1. Der **SETUP-Abschnitt** besteht aus einer Transaktion.
2. Der **DATEN-Abschnitt** kann wegfallen. Er besteht aus keiner, einer oder mehreren Transaktionen.
3. Der **STATUS-Abschnitt (Handshake)** besteht aus einer Transaktion.

Die SETUP-Abschnitt (1 Transaktion)

Die Anfrage (request) erfolgt in der Setup-Token. Der PC sendet das Setup-Token und das Setup-Paket (Daten-Paket). Das Gerät antwortet mit ACK.



Alle Informationen zur Anfrage befinden sich im 8 Byte großen Setup-Paket[6]:

Feld	Bytes
bmRequestType	1
bRequest	1
wValue	2
wIndex	2
wLength	2

bmRequestType

Bit	7	6	5	4	3	2	1	0
	DIR	TYPE1	TYPE0	REC4	REC3	REC2	REC1	REC0

DIR	***Data Phase Transfer Direction***	**0**	**PC zum Gerät (OUT)**
		1	**Gerät zum PC (IN)**

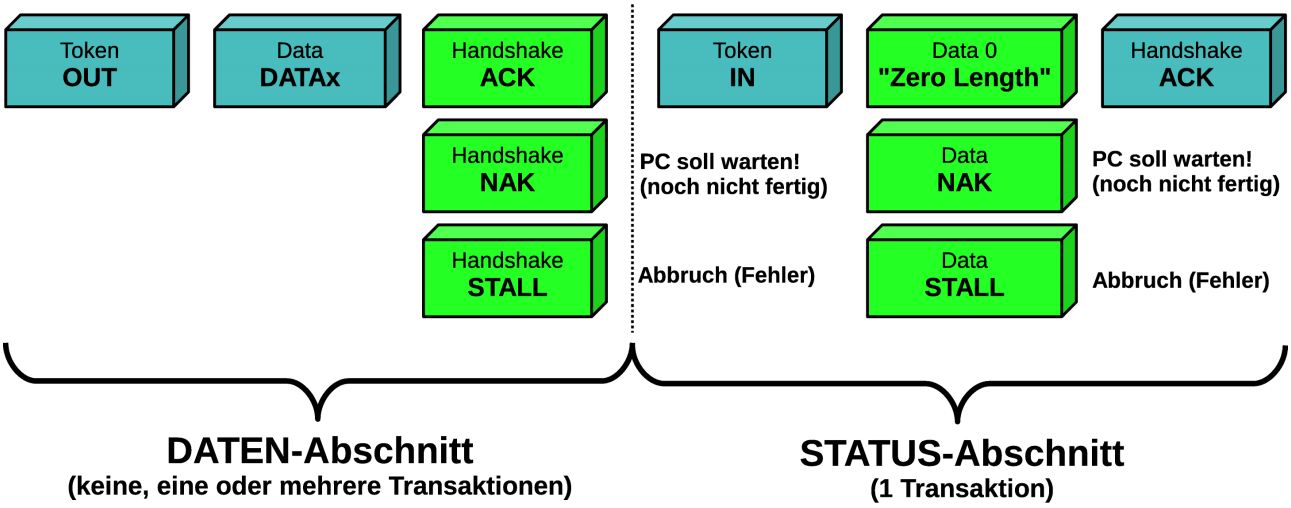
6 Benutzte Präfixe:
b = Byte, w = Word, bm = Bitmap, bcd = binaercodierte Dezimalzahl, i = Index, id = Kennung

TYPE	***TYPE1, TYPE0***	**00**	**Standard**
		01	Klasse
		10	Vendor (herstellerspezifische Anfrage)
		11	Reserviert
REC	***Recipient REC1, REC0***	**00**	**Device**
		01	**Interface**
		10	**Endpoint**
		11	otherReserviert

Der DATEN- und der STATUS-Abschnitt (Handshake)

Der schreibende Control Transfer (Data OUT)

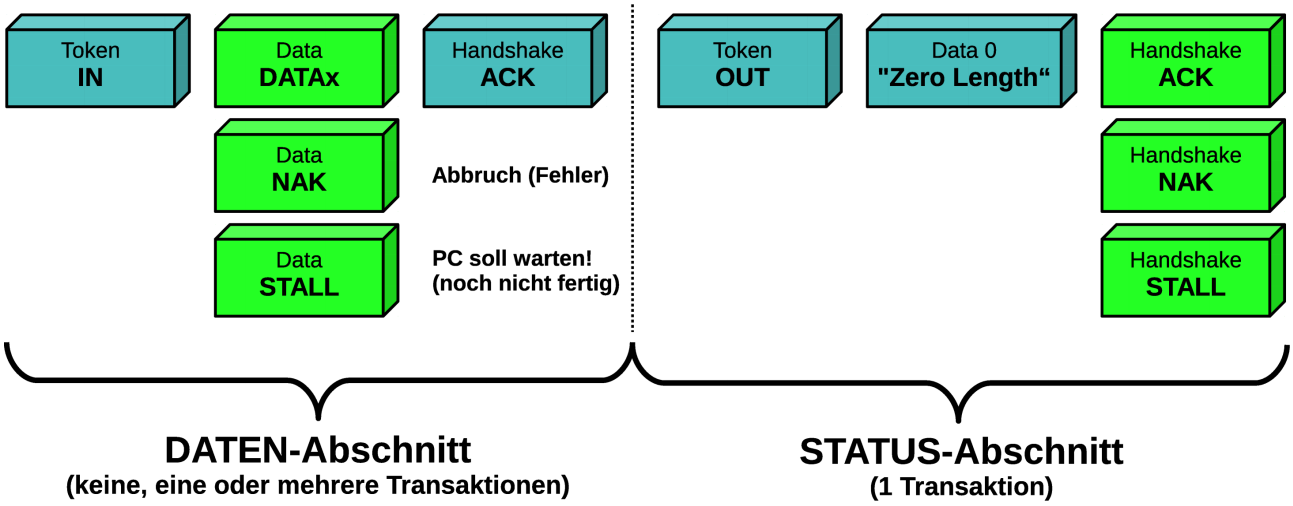
Bei diesem Transfer teilt der PC dem Gerät im Setup-Abschnitt mit, dass er Daten zum Gerät senden will. Bei der Datenphase gibt es nach dem Empfang der Daten dann drei Möglichkeiten. Das Gerät kann den Empfang bestätigen (ACK), den PC auffordern zu warten (NAK) oder Abbrechen (STALL). Tritt beim Token oder Datenpaket ein Fehler auf, so wird das Paket ignoriert. Der DATEN-Abschnitt kann aus mehreren Transaktionen bestehen!



War der Transfer erfolgreich, so sendet das Gerät in der Handshake-Phase ein "Zero Length"-Paket an den PC zurück. Der PC antwortet mit ACK. Trat ein Fehler beim Endpunkt auf, so sendet das Gerät einen STALL. Ist es noch beschäftigt, so sendet es ein NAK. Tritt sonst ein Fehler auf, so wird das Paket ignoriert.

Der lesende Control Transfer (Data IN)

Bei diesem Transfer teilt der PC dem Gerät im SETUP-Abschnitt mit, dass er Daten vom Gerät lesen möchte. Hier gibt es ebenfalls drei Möglichkeiten. Das Gerät kann den IN Token annehmen und die Daten bereitstellen. Der PC antwortet mit ACK. Ist das Gerät noch beschäftigt, so sendet es ein NAK. Trat ein Fehler beim Endpunkt auf, so sendet es einen STALL. Tritt sonst ein Fehler auf, so wird das Paket ignoriert.



In der Handshake-Phase meldet der PC mit einem "Zero Length"-Paket ob er die Daten erfolgreich erhalten hat. Das Gerät antwortet mit ACK. Fehler oder ein beschäftigtes Gerät geben STALL bzw. NAK zurück. Bei sonstigen Fehlern wird das Paket ignoriert.

Die Enumeration

Der Vorgang bei dem der PC dem Gerät eine Adresse zuordnet, alle Informationen über das Gerät erfragt, den richtigen Treiber lädt und dann eine Konfiguration auswählt, wird **Enumeration** genannt.

Mittels Standard-Anfragen (*standard request*) wird vom PC die Adresse des Gerätes festgelegt und mehrere sogenannte Deskriptoren (Beschreibungen) erfragt. Jedes Gerät muss mindestens 4 Deskriptoren liefern:

- **Geräte-Deskriptor (d*evice descriptor*)**
- **Konfigurations-Deskriptor (*configuration descriptor*)**
- **Schnittstellen-Deskriptor (*interface descriptor)***
- **Endpunkt-Deskriptor (*endpoint descriptor)***

Es sollen hier nur die aller nötigsten Schritte einer Enumeration vereinfacht beschrieben werden. Wir gehen dabei von nur einer Konfiguration und einem Interface ohne alternative Einstellungen aus.

Nachdem das Gerät an den Bus angeschlossen wurde, ermittelt der PC anhand der Spannungen der beiden Signalleitungen ob ein Low- oder Full-Speed angeschlossen wurde. Der PC resetet dann das Gerät (beide Datenleitungen auf Low) und stellt dabei fest ob es sich um ein Highspeed-Gerät handelt. Nach dem Reset des Gerätes ist dies bereit über Endpunkt 0 auf der Adresse 0 angesprochen zu werden.

Die Standard Anfragen werden dann mittels SETUP-Paketen (siehe Control Transfer) durchgeführt.

Das erste SETUP-Paket (**Get_Deskriptor**) des PC erfragt (Adresse Null, Endpunkt 0) 64 Byte des Geräte-Deskriptor um die maximale Paketgröße von Endpunkt 0 zu ermitteln[7]. Nach 8 Byte (hier befindet sich die bMaxPacketSize) bricht der PC ab und führt ein neues Reset des Gerätes aus.

Mit dem zweiten SETUP-Paket (**Set_Adress**) sendet der PC eine Geräte-Adresse. Diese wird von der Firmware dem Gerät zugewiesen. Das dritte SETUP-Paket (Get_Deskriptor) erfragt die 18 Byte des Geräte-Deskriptors. Dann werden mit einem vierten SETUP-Paket (Get_Deskriptor) die 9 Byte des Konfigurations-Deskriptors erfragt. Dieses vermittelt die Gesamtlänge des Konfigurations-, Interface- und aller Endpunkt-Deskriptoren. Ein fünftes SETUP-Paket (Get_Deskriptor) erfragt all diese Deskriptoren in einer Aktion. Weitere SETUP-Pakete erfragen optionale Deskriptoren (z.B. String-Deskriptoren).

Der PC lädt jetzt anhand von Vendor-ID und Product-ID und der ".INF" Datei den entsprechenden Gerätetreiber.
Ein letztes SETUP-Paket der Emuneration (**Set_Configuration**) bewirkt das Initialisieren und Aktivieren der User-Endpunkte (Endpunkt1-15). Andere, von einer Firmware nicht unterstützte Anfragen des PCs mittels SETUP-Paketen, werden einfach mit STALL abgewiesen.

Der Geräte-Deskriptor

Es gibt pro Gerät nur einen Geräte-Deskriptor[8].
Hier der Geräte-Deskriptor mit den Werten für unsere minimale Firmware:

Feldbezeichnung	Byte	Wert	Beschreibung
bLength	1	18 (0x12)	Größe des Deskriptors in Byte
bDescriptorType	1	1	Geräte Deskriptor = 1 (Konstante)
bcdUSB	2	0x0110	USB_Spec1_1
bDeviceClass	1	0xFF	Klassencode (hier anbieterspezifisch = 0xFF)
bDeviceSubClass	1	0xFF	Unterklassencode (hier anbietersp. = 0xFF)
bDeviceProtocoll	1	0xFF	Protokollcode (hier anbieterspezifisch = 0xFF)
bMaxPacketSize	1	8	max. Paketgröße Endpunkt 0 (EP0_FS)
idVendor	2	0x03eb	Atmel Code durch usb.org vergeben
idProduct	2	0x0001	Produkt ID beliebig
bcdDevice	2	0x0001	Release Nummer Gerät
iManufacturer	1	1	Index für String-Deskriptor Hersteller
iProduct	1	2	Index für String-Deskriptor Produkt
iSerialNumber	1	3	Index für String-Deskriptor Seriennummer
bNumConfigurations	1	1	Anzahl möglicher Konfigurationen

7 Diese kann 8, 16, 32, 64 Byte betragen.

8 Außer bei Verbundgeräten (*composite device*), die über die Schnittstellen-Deskriptoren statt über den Geräte-Deskriptor beschrieben werden.

Der Konfigurations-Deskriptor

Ein Gerät kann mehrere Konfigurationen besitzen (bNumConfigurations). Ein Gerät kann zum Beispiel eine Konfiguration für die Stromversorgung per Bus besitzen und eine zweite Konfiguration für eine eigenständige Versorgung. Der Gerätetreiber wählt dann die entsprechende Konfiguration aus. Es ist immer nur eine Konfiguration aktiv.

Im Feld **wTotalLength** wird dem PC die Gesamtzahl der Bytes aller Konfigurations-, Schnittstellen- und Endpunkt-Deskriptoren mitgeteilt. In unserem Fall sind das 9 Byte (1 Konfigurations-Deskriptor) + 9 Byte (1 Schnittstellen-Deskriptor) + 3*7 Byte (3 Endpunkte neben Endpunkt 0) = 39 Byte.

Feldbezeichnung	Byte	Wert	Beschreibung
bLength	1	9	Größe des Deskriptors in Byte
bDescriptorType	1	2	Konfigurations-Deskriptor = 2 (Konstante)
wTotalLength	2	39 (0x27)	Länge des Konfigurations-Deskriptors und aller untergeordneter Deskriptoren
bNumInterfaces	1	1	Anzahl der Schnittstellen
bConfigurationValue	1	1	Nummer um diese Konfiguration auszu-wählen (darf nicht Null sein, sonst geht Gerät in den nicht-konfigurierten Zustand)
iConfiguration	1	0	Index für String-Deskriptor dieser Konfiguration (0 = kein Text)
bmAttributes	1	0x80	D7 = 1 Versorgung durch Bus, D6 = 1 Selbstversorgung, D5 = 1 Remote Wakeup
bMaxPower	1	50	Max. Strombezug vom Bus in 2mA Schritten

Der Schnittstellen-Deskriptor

Der Schnittstellen-Deskriptor bündelt mehrere Endpunkte zu einer Funktionsgruppe. Es können mehrere Schnittstellen gleichzeitig aktiv sein (Beispiel, Fax-Schnittstelle, Druck-Schnittstelle und Scan-Schnittstelle bei Multifunktionsdrucker). Eine Schnittstelle kann mit der gleichen Schnittstellennummer mehrere alternative Einstellungen beherbergen, welche allerdings nicht gleichzeitig aktiviert werden können. Ein schnelles Umschalten zwischen den alternativen Einstellungen ist möglich.

Feldbezeichnung	Byte	Wert	Beschreibung
bLength	1	9	Größe des Deskriptors in Byte
bDescriptorType	1	4	Schnittstellen-Deskriptor = 4 (Konstante)
bInterfaceNumber	1	0	Anzahl der Schnittstellen
bAlternateSetting	1	0	Nummer um alternative Einstellungen zu wählen
bNumEndpoints	1	3	Anzahl der Endpunkte außer Endpunkt 0
bInterfaceClass	1	0xFF	Klassencode (hier anbieterspezifisch = 0xFF)

bInterfaceSubClass	1	0xFF	Unterklassencode (hier anbietersp. = 0xFF)
bInterfaceProtocol	1	0xFF	Protokollcode (hier anbieterspezifisch = 0xFF)
iInterface	1	0	Index für String-Deskriptor dieser Schnittstelle (0 = kein Text)

Der Endpunkt-Deskriptor

Die Endpunkt-Deskriptoren beschreiben die Endpunkt (außer Endpunkt 0). Wichtig ist, dass bei der Endpunktadresse auch die Richtung mittels Bit 7 angegeben werden muss.

Hier der Deskriptor für Endpunkt 1 (IN, Bulk, FIFO 8 Byte)

Feldbezeichnung	Byte	Wert	Beschreibung
bLength	1	7	Größe des Deskriptors in Byte
bDescriptorType	1	5	Schnittstellen-Deskriptor = 5 (Konstante)
bEndpointAddress	1	0x81	Bit 7 = 1 (IN), Bit 0-3 Endpunkt-Nummer (andere 0)
bmAttributes	1	2	Transfertyp = Bulk (contr. = 0, iso. = 1, int. = 3)
wMaxPacketSize	2	8	FIFO Größe des Endpunkts in Byte
bInterval	1	0	Polling Interval = 0 (ignoriert für Bulk und Control), 1 für Iso, 1-255 für Interrupt

Die String-Deskriptoren

Sie sind nicht unbedingt nötig, liefern aber zusätzliche lesbare Informationen. Wird ein String-Deskriptor nicht benötigt, so wird sein Index auf 0 gesetzt.

Strings sind in Unicode (16 Bit) kodiert. Es werden viele unterschiedliche Sprachen unterstützt. Im String-Deskriptor mit dem Index 0 werden die unterstützten Sprachen festgelegt. Hier als Beispiel eine Unterstützung für Englisch und Deutsch:

Feldbezeichnung	Byte	Wert	Beschreibung
bLength	1	6	Größe des Deskriptors in Byte
bDescriptorType	1	3	String-Deskriptor = 3 (Konstante)
wLANGID[0]	2	0x0409	English USA (Standard)
wLANGID[1]	2	0x0407	Deutsch Standard

Als nächster Deskriptor folgt dann der Deskriptor mit Index Eins (hier Hersteller String-Deskriptor)

Feldbezeichnung	Byte	Wert	Beschreibung
bLength	1	18	Größe des Deskriptors in Byte
bDescriptorType	1	3	String-Deskriptor = 3 (Konstante)
bString	16	0x0057,0x0045,.....	"W","E","I","G","U",".","L","U"