Erledigt "Du hast ja Alles" - hmmm vielleicht, wenn ich einen Laptop habe.

Beitrag von „Brumbaer“ vom 12. April 2018, 11:13

Nach Hause telefonieren
Wollen wir das nicht alle ?
Na ja, vielleicht, aber ohne leuchtenden Finger, es sei denn wir wären süd-koreanischen Haustierklone.
Eine ständige Internetverbindung all-überall ist was tolles. Ich habe mich an das LTE Modem im iPad so gewöhnt, dass kein Gerät ohne LTE Modem das iPad ersetzen kann.
Das eingebaute LTE Modem war einer der Gründe für die Wahl des Miix.

Alles ganz einfach
Wo habe ich das schon gehört ?
Egal - IORegistryExplorer zeigt unter den USB Geräten das Modem an.
Typ L831-EAU Hersteller *Fibocom* - und jetzt kommt's - *IORegistryExplorer* zeigt an, dass "Freude schöner Götter Funken", Treiber geladen werden.

Ein Gefühl, wie bei km 42 eines Marathonlaufes. Nicht, dass ich jemals einen Marathon gelaufen wäre oder es in Betracht ziehen würde oder es in Betracht ziehen würdew es in Betracht zu ziehen, aber so muss es sich anfühlen, körperlich und emotional erschöpft, aber das Ziel in Sicht, ein letztes Mobilisieren der Kräfte und dann ins Ziel.
Habe ich auch verdient nach dem Stress mit dem Miix. Einen Tee gekocht, eine Box von nem iPad gesucht, den SIM-Karten-Schlitten-Auswurf-Stecker - Stecker und Auswurf passt nicht ? Doch, doch. Man muss was reinstecken, damit was raus kommt, wie beim Kaugummi-Automaten. Liebe Kinder, Kaugummi-Automaten sind ... googelt's halt.

Ist denn schon Ostern

Wo war ich ? Ach ja das Dingens gepackt und die SIM Karte aus dem iPad genommen. Jetzt muss man nur noch den Steckplatz am Miix finden. Wenn das Teil mit ausgeklapptem Ständer vor einem steht, sieht man ihn erst mal nicht. Ich sehe eh besser mit den Händen, also die Seiten abgetastet - nichts. Also gut Gerät von der Tastatur abgezogen, Ständer rangeklappt. Miix nach links, rechts, oben und unten gedreht und nichts. Wenn das so weiter geht, muss ich ins Handbuch schauen. "Nicht mit mir", denke ich, "ich der Mann, der ein Bit am Geruch erkennt, den noch kein Byte gebissen hat, der morgens mit dem Datenbus zur Arbeit fährt und abends mit dem Adressbus nach Hause kommt, der soll in ein Handbuch schauen ? Haaahhh, niemals". Die Schnellstartanleitung tut's auch.
Es ist auch ganz einfach, man klappt den Ständer auf und dann sieht man dahinter zwar nicht den SIM Karten Slot, aber immerhin das Logo des Slots. Also mal getastet und gleich gefunden. Ich sach ja, ein Fachmann wie ich, findet so einen Slot ja sofort. Der braucht kein Handbuch.
Die Karte geht etwas schwer rein. Vermutlich weil sie aus einer größeren Karte gestanzt wurde und etwas dicker ist, als die neuen Nanos. Selbstoptimierung macht auch vor SIM Karten nicht halt. Wahrscheinlich ernähren die sich auch nur von veganem Strom.

Fasching ist draußen

Auf den Straßen herrscht buntes Treiben, aber hier tut sich nichts. Also Windows gestartet und das Modem getestet und was soll ich sagen ? Es läuft - zumindest das Modem.
Zurück zu MacOS. Offensichtlich wird ein Ethernettreiber geladen, es kommen nur keine Daten, vermutlich besteht keine LTE-Verbindung.

Mühsam

Die Suche nach Informationen über das Modem führt Widersprüchliches zu Tage. In der Produktliste taucht es erst gar nicht auf und die Handbücher, die Google findet, beschreiben unterschiedliche Versionen.
Wahrscheinlich heißt es Handbuch, weil man trotz des Buches, alles von Hand rauskriegen muss.

Klassisch

Klassischerweise hat ein Modem zwei Komponenten. Die eine für die reine Datenübertragung und die andere für den Verbindungsaufbau. Die für Datenübertragung tut beim Mac so als wäre sie ein *Ethernetcontroller* und die andere als sei sie ein serielles *AT Modem*. Den *AT Befehlssatz* gibt es seid Ewigkeiten. Inzwischen wurde er natürlich, um ein paar Features z.B. für den Verbindungsaufbau über Handy Netz erweitert. Bei den Erweiterungen kocht zum Teil jeder Hersteller seinen eigenen Buchstabensalat.
Bei der Suche nach Informationen über das *L831-EAU* bin ich auch über ein Handbuch zu dessen Interpretation des AT Chipsatzes gestolpert.

Desinteresse
Schaut man sich obigen Auszug aus dem *IORegistryExplorer* an, sieht man

- *HS03*. Das ist der USB Port - interessiert uns nicht.
- *L831-EAU@14300000* ist das USB-Gerät. Gaaanz toll - aber interessiert uns nicht. Dann sehen wir zwei Treiber.
- *AppleUSBHostCompositeDevice* ist ein Treiber für USB Geräte, die mehrere Funktionen in sich vereinen. Welche Funktion ein USB Gerät hat geben seine *Klasse, Unterklasse und Protokoll* an. Das Tripel sagt zum Beispiel *Drucker*. Und für diese Funktion gibt es dann ein *Interface*. Manchmal hat ein Gerät aber mehr als eine Funktion, z.B. *Drucker und Scanner.* Dann hat man ein *Composite Device* und Geräte Klasse, Unterklasse und Protokoll verweisen bezüglich der Funktion auf die Interfaces. Und es gibt ein Interface für die Druckfunktion und eins für die Scanfunktion, jeweils mit passender Klasse, Unterklasse und Protokoll. *AppleUSBHostCompositeDevice* verwaltet solche Mehrfach-Funktionsgeräte und ist im laufenden Betrieb eher unsichtbar. Kurzum interessiert hier nicht.
- *AppleUSBHostLegacyClient* stellt die Kompatibilität mit älteren Programmen her. Interessiert hier auch nicht.

Desdesinteresse
Auf der selben Ebene gibt es noch *Data (OFF)@1* und Fibocom *L831-EAU (NCM)@0*. Dies sind die zwei Interfaces mit ihren Treibern. Das Data Interface lädt einen Ethernettreiber. Das sieht gut aus, aber das andere, zeigt keinen seriellen Treiber, also klappt die normale Methode mit AT Befehlen über eine serielle Schnittstelle nicht.

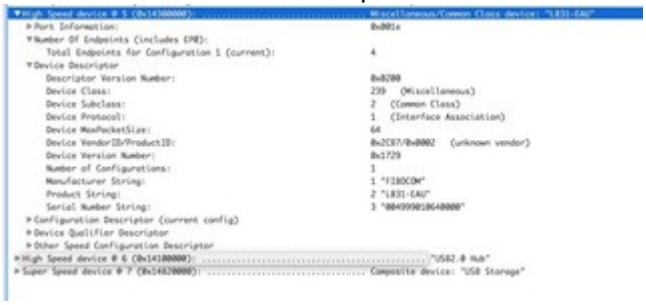
Bestandsaufnahme
Da stellt sich uns die Frage, "was gibt's zu Mittag ?". Nein, also doch schon, aber eigentlich, wollte ich auf "Hat das Modem eine serielle Schnittstelle und wenn nein was hat es stattdessen ?" hinaus.

Statt nur das Ergebnis zu präsentieren, zeige ich im Folgenden wie das Ergebnis zu Stande kam. Wenn man schon weiß wie es geht, ist es in einer Minute erledigt. Dank der Erklärung dauert es deutlich länger. Die, die USB Deskriptoren und der Aufbau von USB Geräten nicht interessiert oder ihn schon kennen, können gerne zu *Was sagt uns das ?* springen.

USB Stocherer
Wieder mal eine schöne Übersetzung. *USB Prober* ist eine App und Teil des *Apple USBFamily Log Paketes*. Es soll Hardware Entwickler beim Anbinden von USB Geräten und der USB Treiber Entwicklung unterstützen. Dass die neueste Version zum System 10.9.4 gehört, sagt viel über den aktuellen Stand der Unterstützung von Hardware Entwicklern. Ob die Unterstützung für

große Firmen besser ist, weiß ich nicht.
Ok, genug gejammert, der Miix liefert genug Gründe zum Jammern, da brauch ich nicht noch Apple für.
Ach ja, das Paket findet man auf der Apple Webseite unter https://developer.apple.com/download/more/. Im Suchfeld USB eingeben. Ob man es mit jeder Art von Entwickleraccount runter laden kann oder ob es noch sonst wo zum Download angeboten wird weiß ich nicht.

Was haben wir denn da ?
Die USB Spezifikation legt fest, dass jedes USB Gerät über *Descriptors* definiert wird. Ein Descriptor beschreibt die Eigenschaften einer Komponente und ihre Unterkomponenten. Die Deskriptoren bilden so eine Hierarchie in Form einer Baumstruktur. Die Struktur der Deskriptoren spiegelt die Struktur der Komponenten eines USB Gerätes wieder. Am Ende des Textes ist eine Grafik, die die Organisation der Deskriptoren nicht als Baum, sondern als Schachtelbild zeigt.
USB Prober liest die Deskriptoren und bereitet sie in von Menschen lesbarer Form auf:



In der ersten Zeile erfahren wir, dass es sich um das *fünfte USB Gerät* handelt, über *USB 2.0* (High Speed) kommuniziert und am Port 3 (die 3 in *0x14300000*) hängt, der Klasse *Verschiedenes* angehört und das Gerät *L831-EAU* heißt.
Uns interessiert das Gerät, nicht wo am Computer es angeschlossen ist, deshalb ignorieren wir die Port Informationen.

Das Ende ist nah
Endpoints sind die Punkte an denen die *Pipes* anknüpfen. *Pipes* sind die Datenübertragungs-Kanäle. Die Daten fließen durch die *Pipes* von einem *Endpoint* zum Computer oder vom Computer zu einem *Endpoint*.

Es geht nicht ohne
Die Geräteverwaltung hat einen eigenen Endpoint, den *Endpoint 0*. Das ist der *Control Endpoint* . Computer und USB kommunizieren nach einem strengen Protokoll. Der Computer sendet

einen 8 Byte langen *Request*. Danach können noch Daten folgen, die vom Computer oder dem USB-Gerät stammen können. Es gibt vordefinierte *Requests* für alles Mögliche und man kann auch eigene *Requests* definieren. Zu den vordefinierten *Requests* gehören zum Beispiel das Wählen einer Konfiguration und das Abfragen der Deskriptoren. Ohne diese Fähigkeiten geht nichts, deshalb muss jedes USB Gerät einen Endpoint 0, einen *Control Endpoint* haben.

Aber es geht auch anders

Der *Control Endpoint* wird mit dem Gerät selbst und dessen Verwaltung assoziiert und sein Übertragungsprotokoll ist nicht wirklich effizient.

Deshalb können *Interfaces* (kommt gleich) eigene Endpoints definieren.
Diese Endpoints können nur gelesen oder beschrieben werden, aber im Gegensatz zum Endpoint 0 nicht beides.
Soll ein Interface Daten empfangen und senden können, muss es zwei Endpoints bekommen, einen zum Senden und einen zum Empfangen.

Endpoints werden durchnummeriert. Da die 0 schon vom Control Endpoint belegt ist, beginnen die Endpoints, die beschrieben werden, bei der 0x01 und Endpoints, die gelesen werden, bei 0x81.
Die Datenübertragung erfolgt nicht mit Requests, sondern in einem von drei anderen Modi:

- *Bulk* - Byte folgt auf Byte, Computer kontrolliert, z.B. Daten von oder zu einer Festplatte
- *Interrupt* - Das Gerät schickt Daten unaufgefordert an den Computer. Falls das Gerät dem Computer was sagen muss. Z.B. bei einem Überwachungsgerät oder wenn das Gerät was macht und sich dann meldet wenn es fertig ist. In Wahrheit fragt der Computer regelmäßig nach, ob das Gerät Daten hat, die es loswerden will. Das passiert aber im verborgenen und sieht von außen so aus als wäre da ein Interrupt am Werke.
- *Isochronous* - Daten werden als kontinuierlicher Datenstrom übertragen. Z.B. von einer Videocamera. Wie bei einem Hop-on Bus, kann mann jederzeit ein- oder aussteigen.
- Wie gesagt ein USB-Gerät muss über keinen solchen Endpoint verfügen, aber es muss einen Control-Endpoint besitzen.

Der L831-EAU hat laut USB Prober in der momentan ausgewählten Konfiguration 4 Endpoints inkl. des Endpoints 0. Der Endpoint wird vom Gerät zur Verfügung gestellt. Die anderen 3 Endpoints müssen von einem oder mehreren Interfaces zur Verfügung gestellt werden.
Die Anzahl und Art der Interfaces und damit die Anzahl und Art der Endpoints hängt von der jeweiligen Konfiguration (*Configuration*) ab. Es ist selten, dass eine Gerät mehr als eine Konfiguration unterstützt. USB Prober weißt auch nur einen *Configuration Descriptor* aus. D.h. unser Gerät hat auch nur eine Konfiguration.
Die schauen wir uns später an.

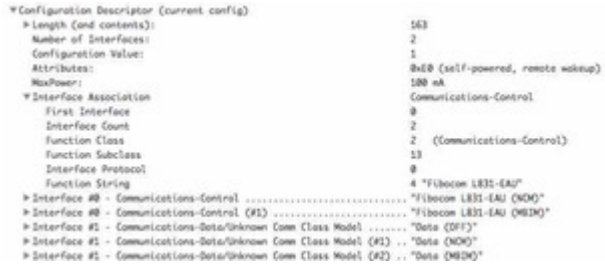
Der *Device Descriptor* erzählt uns Allgemeines über unser USB Gerät. Die meisten der Einträge sollten selbst erklärend sein. *Number of Configurations* bestätigt uns, dass es nur eine Konfiguration gibt.
Device Class, Subclass und Protocol verdienen besondere Erwähnung. USB verwendet Klassen, Unterklassen und Protokolle, um Geräte in Gruppen einzuteilen. Alle Geräte einer Gruppe haben bestimmte Eigenschaften. Typische Gruppen sind *Drucker*, *Audio/Video* oder *Drahtlos*. Unter http://www.usb.org/developers/defined_class/#BaseClassEFh findet man eine Auflistung. In unserem Falls ist die *Klasse 239*. Das ist wenig hilfreich, steht die 239 doch für *Verschiedenes*. Unterklasse 2 und Protokoll 1 bedeuten "es gibt einen *Interface Association Descriptor*, der mehr über die Geräte Funktion aussagt". Dieser ist Teil der Konfiguration, wir werden ihn deshalb später als Teil des *Configuration Descriptors* wiederfinden.

Das Leben der anderen

Den *Configuration Descriptor* schieben wir noch mal nach hinten. Und so kommen wir zu *Device Qualifier Descriptor* und *Other Speed Configuration Descriptor*. Beide gibt es nur bei *High Speed* (USB 2.0) Geräten. *High Speed* Geräte unterstützen zwei Geschwindigkeiten: *Full-(USB 1.0) und High-Speed (USB 2.0)*. In der ersten Zeile stand, dass im Moment High Speed verwendet wird. Nun ist es möglich, dass sich die Deskriptoren im High- und Full-Speed Modus unterscheiden. Diese beiden Deskriptoren beschreiben das Gerät im "anderen", dem gerade nicht verwendeten Modus. Es ist uns egal was im Full-Speed Modus wäre, wir werden ihn nie verwenden, deshalb ignorieren wir die beiden Deskriptoren.

Last, not least

Jetzt können wir uns nicht länger vor dem Configuration Descriptor drücken.

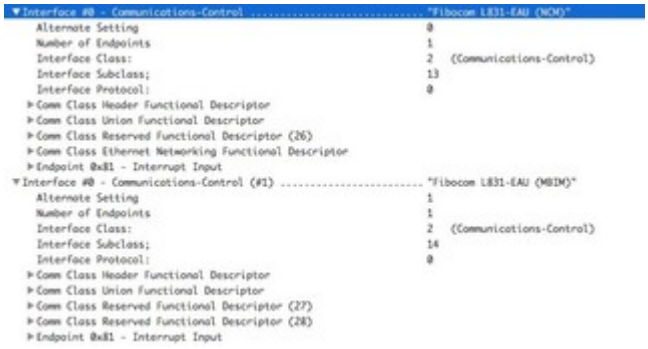


Der *Configuration Descriptor* trägt selbst wenig zur Erleuchtung bei, aber seine Unterdeskriptoren haben es in sich.
Bei einem USB Gerät stellen die *Interfaces* die Funktionen zur Verfügung. Bei einer Drucker, Scanner, Fax Kombi würde man je ein *Interface* für die Druck, Scan- und Faxfunktion anlegen. Man kann auch Alles irgendwie in einem *Interface* verwurschteln, aber vorgesehen ist eine Aufteilung der Funktionen auf verschiedene *Interfaces*.
Der *Interface Association Descriptor* gibt ergänzende Informationen über die Konfiguration und das Zusammenspiel der *Interfaces*. Er ist nur notwendig, wenn im Device Descriptor auf ihn

verwiesen wird.
Das erste Interface (*First Interface*) ist das Interface mit der Nummer 0. Insgesamt (*Interface Count*) gibt es zwei Interfaces. Der eine oder andere mag sich erinnern, dass der Device Descriptor sich nicht auf eine Funktion festnageln lassen wollte und auf den *Interface Association Descriptor* verwiesen hat. Und der sagt jetzt Klasse 2 (Kommunikationsgerät) , Unterklasse 13 (NCM - Network Control Model) und Protokoll 0 (No encapsulated commands / responses.).
Also ein Modem, dass der NCM Unterklasse angehört und keine encapsulated commands/respones unterstützt.

Die fünf, die zwei waren

Anzahl der Interfaces ist laut Configuration Descriptor 2, aber wir sehen 5 Interfaces. Na ja es sind nur 2 Interfaces, Interface 0 und Interface 1, von denen gibt es aber 2 bzw. 3 Varianten. Wozu denn das, warum hat man nicht einfach zusätzliche Konfigurationen angelegt ?
Eine Konfiguration, kann den Charakter des Gerätes komplett ändern und wenn man zwischen zwei Konfigurationen wechselt ist es so als ob man das Gerät abzieht und ein anderes Gerät anschließt. Alles auf Anfang. Schaltet man bei der Druck-, Scan-, Fax-Kombi die Konfiguration um, um z.B. die Drucker Emulation zu wechseln, brechen auch laufende Scans und Faxübertragungen ab und auch Scanner und Fax, werden zurückgesetzt.
Das Umschalten auf ein A*lternatives-Interface*, betrifft nur dieses Interface und lässt den Rest des Gerätes unberührt. Das erlaubt ein relative glattes Umschalten einzelner Funktionen im laufenden Betrieb.



Die zwei *Alternativen* für das Interface 0 sind sich ziemlich ähnlich. *Alternate Setting* gibt an um welche Alternative es sich handelt. Alternative 0 ist die, die nach einem Neustart automatisch gewählt wird. Beide Alternativen haben einen *Interrupt Endpoint* mit der Nummer 0x81. Da die Nummer mit 8 beginnt handelt es sich um einen Endpoint der gelesen wird. *Interface Class, Subclass* und *Protocol* definieren wieder die Art des Interfaces. Bei der ersten Alternative handelt es sich um die uns aus dem *Interface Association Descriptor* vertrauten Werte. Alternative 1 ist ein Kommunikationsgerät, Unterklasse MBIM (Mobile Broadband Interface Model) und Protokoll MBIM-Protokoll.
Die Gruppenzugehörigkeit beeinflusst die verfügbaren *Functional Descriptors*. Diese

beschreiben verschiedene Aspekte dieser speziellen Interface Art und interessieren uns in diesem Moment nicht.
Lange Rede stark gekürzt. Interface 0 kontrolliert die Kommunikation nach NCM oder MBIM. Was auch immer das sein mag.

Interface 1 hat 3 Varianten.
Alternative 0 hat keine Endpoints, macht demzufolge Datenübertragungs-technisch nichts, also *Datenübertragung aus*. Die anderen beiden Varianten haben jeweils einen *Lese-* und *Schreib-Endpoint* vom Typ *Bulk*. Also massenweise Daten rein und raus - macht Sinn.
Alle drei gehören der Datenübertragungsklasse, Abteilung Daten an. Interface 0 war Abteilung Kontrolle. In der Abteilung Daten sind keine Unterklassen definiert, weshalb Subclass bei allen dreien auf 0 steht. Das Protokoll der ersten beiden Alternativen ist *Networktransport* Block bei der letzten Alternative *MBIM*.
Interface 3 bietet also drei Alternativen, Aus, NCM Daten und MBIM Daten.

Einen hab ich noch

Der Vollständigkeit halber schauen wir uns noch einen *Endpoint Descriptor* an. Wir sehen seine Nummer (*Address*), den Übertragungsmodus (*Attributes*), die maximale Größe eines Paketes (*Max. Packet Size*) und wie oft nach Daten geschaut werden soll, falls es sich um einen Interrupt Endpoint handelt (*Polling Interval*).
Da die Endpoint Descriptors zu den Interfaces gehören, gibt es keinen Deskriptor für den Control Endpoint der gehört ja zum Gerät. Auf der anderen Seite, seine Adresse ist immer 0, sein Übertragungsmodus immer Control und abgefragt, wird er auch nicht. Also bleibt nur die maximale Paketgröße. Die findet sich tatsächlich oben im *Device Descriptor* als *Device MaxPacketSize*. Alles gut.

Sag mir wo die Texte stehen, wo sind sie geblieben ?

Texte haben unterschiedliche Längen und können in verschiedenen Sprachen vorliegen. Damit die einzelnen Deskriptoren eine feste Größe haben und man mit wenig Aufwand mehrere Sprachen unterstützen kann, werden die Texte nicht in den Deskriptoren gespeichert.

Stattdessen wird nur mittels einer Nummer (*index*) auf einen String Deskriptor verwiesen. Dieser String Descriptor enthält dann den Text.
USB Prober ersetzt die Nummern in den Deskriptoren freundlicherweise gleich durch die entsprechenden Texte.

Was sagt uns das ?
Das Gerät dient der Datenübertragung und unterstützt die Datenprotokolle NCM und MBIM.
Zu jedem gibt es eine passendes Daten- und ein passendes Kontrollinterface. Beide Kontrollinterfaces haben jeweils nur einen Interrupt-Lese-Endpoint. Das ist also bestimmt keine serielle Schnittstelle.

Also kein Umschalten der Konfiguration oder Umschalten auf ein alternatives Interface und schon ist alles wie wir es gerne hätten. Wir werden mit den Standardmethoden und AT Befehlen, nicht weiterkommen.
Also zurück zum Anfang. Im IORegistry Explorer stand ja auch, dass kein serieller Treiber für das Kommunikations-Kontroll-Interface geladen wurde, sondern AppleUSBNCMControl.

Denen, die finden für diese Erkenntnis hat sich der Exkurs in die Welt der Deskriptoren nicht gelohnt, sei gesagt: They will be back.

Wenn das km 42 ist, dann liegt das Ziel hinter einer Kurve, denn zu sehen ist es noch nicht.

PS Descriptor