Komunikace na Univerzální sériové sběrnici

Martin Fridrich

Bakalářská práce
2021

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin Fridrich**
Osobní číslo:	**A16436**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Komunikace na Univerzální sériové sběrnici (USB)**
Téma práce anglicky:	**A Universal Serial Bus (USB) Communication**

Zásady pro vypracování

1. Popište sběrnici USB a principy komunikace.
2. Otestujte softwarové nástroje pro zachycení a analýzu komunikace.
3. Porovnejte možnosti SW nástrojů a vyberte vhodné nástroje pro použití v laboratoři.
4. Připravte vzorové úlohy pro odesílání a příjem dat po USB.
5. Navrhněte laboratorní úlohu, ve které propojíte teoretické znalosti a praktické zkušenosti.

Forma zpracování bakalářské práce: **Tištěná/elektronická**

Seznam doporučené literatury:

1. GOOK, Michael. Hardwarová rozhraní: průvodce programátora. Brno: Computer Press, 2006. Hardware (Computer Press). ISBN 80-251-1019-2.
2. MUELLER, Scott. Upgrading and Repairing PCs. 21st Edition. Pearson Education, Inc. / Que Publishing. ISBN 978-0-7897-5000-6.
3. USB Implementers Forum. Document Library [online]. USB-IF, 2019 [cit. 2019-11-11]. Dostupné z: https://www.usb.org/documents
4. WIRESHARK FOUNDATION. Wireshark User's Guide. Wireshark [online]. [cit. 2019-11-18]. Dostupné z: https://www.wireshark.org/docs/wsug_html_chunked/
5. USB IMPLEMENTERS FORUM. Universal Serial Bus (USB): HID Usage Tables [online]. 2004 [cit. 2019-11-19]. Dostupné z: https://www.usb.org/sites/default/file/documents/hut1_12v2.pdf

Vedoucí bakalářské práce: **doc. Ing. Martin Sysel, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **15. ledna 2021**
Termín odevzdání bakalářské práce: **17. května 2021**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

prof. Mgr. Roman Jašek, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 10. 5. 2021

Martin Fridrich, v. r.
podpis studenta

ABSTRAKT

Tato bakalářská práce se zabývá způsobem komunikace na USB. V teoretické části se práce zaměří, co to USB vlastně je, jak vypadá a jak funguje. Ve fyzické vrstvě práce popisuje existující konektory a používaný kabel, u elektrické vrstvy popisuje napájení při přenosu dat. Následně obsahuje princip datové komunikace (její topologii, typy přenosů atd.) a představí několik softwarových snifferů použitých pro odposlech USB komunikace pro praktickou část. V praktické části je cílem vytvořit vzorové úlohy a následně z nich vytvořit laboratorní úlohu sloužící ke snadnějšímu pochopení způsobu komunikace USB.

Klíčová slova: USB, sniffer, Wireshark

ABSTRACT

This bachelor thesis deals with the USB communication. The theoretical part of the thesis focuses what the USB really is, how it looks and how it works. In a physical layer the thesis describes existing connectors and connecting cable. In an electrical layer it describes a power during data transmission. Subsequently, it explains the USB principle of data communication (its topology, transmission types, etc.) and introduces several software sniffers used for sniffing of the USB communication which it is used in the practical part. The objective of practical part is to create sample tasks and then to create a laboratory task to facilitate understanding of the USB communication.

Keywords: USB, sniffer, Wireshark

Děkuji doc. Ing. Martinu Syslovi, Ph.D. za odborné vedení práce a za jeho nekonečnou trpělivost.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Od počátku osobních počítačů vzniklo bezpočet různých periferií s různými sběrnicemi a před vznikem USB se jich používala široká škála. Každý druh periferie často vyžadoval různé typy sběrnic a někteří výrobci periferií si dokonce vytvořili sběrnice i své vlastní. S přibývajícími počty sběrnic, se začaly ozývat hlasy po sjednocení a zjednodušení, které zapříčinilo vzniku USB. [1]

Univerzální sériová sběrnice je technický standard vydaný roku 1996 jako USB 1.0, zastřešený sedmi předními výrobci periferií a počítačových komponentů: Compaq, DEC, IBM, Intel, Microsoft, NEC a Nortel. Jeho cílem byla potřeba standardizovat sběrnici mezi periferiemi a osobním počítačem a vytvořit tak univerzální řešení pro přenos informací nevyhnutelné v budoucím vývoji počítačových technologií. K většímu rozšíření USB došlo až s vydáním USB 1.1 roku 1998, kdy jej Apple implementoval do svého iMacu a pak následovali i další výrobci počítačů. V roce 2000 už došlo k masovému rozšíření, když vyšla nová verze USB 2.0 zejména díky datovým nosičům a dalším standardizovaným modifikacím konektoru. S příchodem USB 3.0 v roce 2008 došlo k výraznějším změnám zejména na fyzické vrstvě a navýšení rychlosti přenosu dat. Tato verze prošla 2 revizemi, kde došlo ke změnám datové rychlosti a dalším drobným úpravám. První revize jako USB 3.1 vyšla v roce 2013, druhá revize USB 3.2 pak v roce 2017 a přišla navíc s novým konektorem USB-C. Zatím poslední verze USB4 vyšla v roce 2019. [2] [3] [4]

USB za svou dobu existence nahradila většinu počítačových sběrnic a je pravděpodobně nejrozšířenějším sběrnicí vůbec. Vzhledem k dalšímu vývoji nových generací USB, bude tento trend prozatím pokračovat. [5]

Mezi hlavní důvody, které zapříčinily úspěch USB, patří:

- automatická konfigurace ovladačů tzv. Plug and Play,
- jedná se o levné řešení s dostatečně vysokým datovým přenosem (od USB 2.0 i pro podporu přenosu dat pro hlas, zvuk nebo video v reálném čase),
- různé varianty konektorů,
- možnost připojit a odpojit zařízení bez nutnosti restartu systému tzv. hot swapping,
- v jednu chvíli je možné připojit až 127 zařízení na jeden rozbočovač,

- zpětnou kompatibilitu pro předchozí generace USB. [5]

Právě kvůli tomuto velkému rozšíření sběrnice si tato práce klade za cíl, popsat komunikaci USB pro možné využití ve výuce.

V teoretické části bude popsána architektura USB, jak probíhá přenos informací, mechanické a elektrické vlastnosti.

V praktické části na základě informací z teoretické části bude otestováno několik softwarových nástrojů pro zachycení USB komunikace a jejich porovnání. S použitím těchto nástrojů vzniknou vzorové úlohy pro odesílání a příjem dat. Z úloh potom bude vytvořena laboratorní úloha použitelná pro edukativní účely k pochopení principů komunikace USB.

I. TEORETICKÁ ČÁST

1 POPIS SYSTÉMU A VLASTNOSTÍ USB

Tato kapitola představuje stručný úvod na celou architekturu a její klíčové vlastnosti. Některé části důležité pro USB komunikaci mohou být rozšířeny v dalších kapitolách.

USB je sběrnice, která umožňuje výměnu dat mezi hostitelským počítačem a širokou škálou zařízení. Taková komunikace může být připojení zařízení, její konfigurace, datový přenos a odpojení zařízení. Připojené zařízení mají sdílenou šířku pásma přes token-based protokol řízený hostitelem. [6]

USB systém se dělí na 3 části [6]:

- USB hostitel (host)
- USB zařízení (device)
- USB propojení (interconnect)

USB propojení je způsob fyzické a logické komunikace mezi zařízením a hostitelem. Jsou to tyto druhy propojení [6]:

- Topologie směrnice
- Komunikace mezi vrstvami
- Datový přenos na protokolové úrovni

1.1 USB hostitel

Hostitel zahajuje všechny komunikace na sběrnici. USB rozhraní k hostitelskému systému se nazývá hostitelský řadič. V hostitelském systému je integrován kořenový rozbočovač. Hostitel je v každém USB systému pouze jeden. [6]

1.1.1 USB hostitel: hardware a software

Hostitel řídí jiná USB zařízení přes hostitelský řadič. Ten řídí [6]:

- Detekci připojení a odpojení zařízení
- Správu řídicích toků mezi hostitelem a zařízeními
- Správu datových toků mezi hostitelem a zařízeními
- Shromažďování statistik o stavu a činnosti
- Poskytnutí napájení připojeným zařízením

USB systémový software na hostiteli spravuje interakce mezi zařízeními a softwarem zařízení založený na hostiteli. Tyto interakce jsou [6]:

- Rozpoznání a konfigurace zařízení
- Izochronní datové přenosy
- Asynchronní datové přenosy
- Správa napájení
- Správa informací zařízení a sběrnice

1.2 USB zařízení

Všechna zařízení jsou upstream připojení, což znamená, že datový tok směřuje od zařízení k hostiteli. Downstream pak znamená, že datový tok míří od hostitele k zařízení. Upstream a downstream konektory nejsou mechanicky zaměnitelné, čímž se vyloučí loopback připojení v rozbočovačích. [6]

USB zařízení se dělí na [6]:

- Koncová zařízení, které poskytují další rozšířené možnosti systému, jako například klávesnice, myš atd.
- Rozbočovače, které umožňují další připojení koncových zařízení

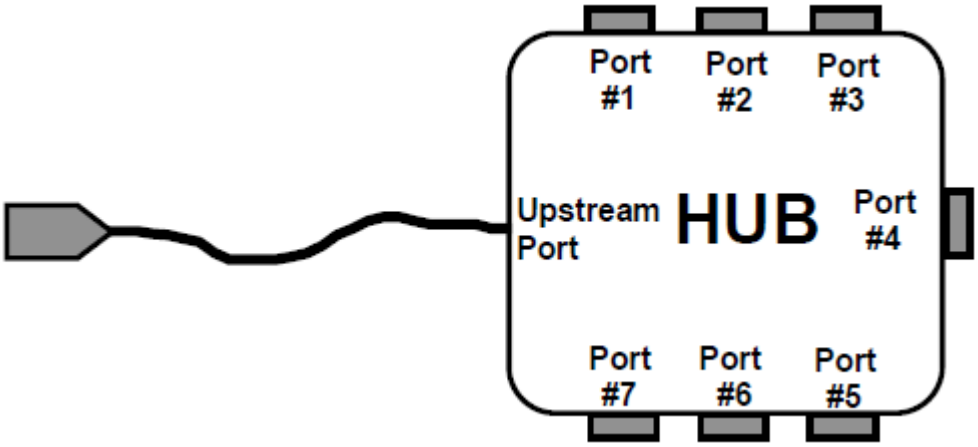
1.2.1 Rozbočovač

Rozbočovače (Hub) jsou klíčovým prvkem pro plug-and-play. Slouží k zjednodušení konektivity a poskytují robustnost za relativně nízké náklady a komplexitu. [6][9]

Rozbočovače jsou zařízení umožňující vícenásobná připojení USB přes body připojení, které se pak nazývají porty. Každý rozbočovač může vytvořit z jednoho bodu připojení více bodů připojení, protože architektura umožňuje tyto rozbočovače řetězit. Rozbočovač má jeden upstream port a vícero downstream portů. Upstream port připojuje rozbočovače k dalším rozbočovačům nebo přímo k hostiteli. Downstream port umožňuje připojit další rozbočovač nebo koncové zařízení. Rozbočovače umí detekovat u každého svého downstream portu připojení/odpojení a umožňují distribuci energie dalším zařízením. Každý downstream port může běžet samostatně jinou rychlostí dat. [6]

Rozbočovač se skládá z 3 částí: řadiče (Hub Controller), opakovače (Hub repeater) a překladače transakcí (Transaction Translator). Opakovač je protokolově řízený přepínač

mezi upstream a downstream portem. Má také hardwarovou podporu pro reset a sunspend/resume signalizaci. Hostitelský řadič poskytuje komunikaci od nebo k hostiteli. Specifické stavové a řídící příkazy rozbočovače umožňují hostiteli rozbočovač konfigurovat a ovládat tak jeho porty. Překladač transakcí poskytuje mechanismy umožňující podporu zařízení nižších rychlostí. [6]



Obr. 1. Schéma rozbočovače [6]

1.2.2 Koncové zařízení

Koncové zařízení (Function) je USB zařízení, které vysílá, přijímá nebo řídí informace na sběrnici. Buď se jedná o samostatné periferní zařízení s kabelem, které se připojí k portu na rozbočovači nebo o složené zařízení, kde je možné mít více koncových zařízení s vlastním integrovaným rozbočovačem a jedním kabelem v jednom v zařízení. Hostiteli se pak jeví jako rozbočovač s jedním nebo více neodnímatelnými zařízeními. [6]

Každé koncové zařízení obsahuje informace o konfiguraci popisující možnosti zařízení a jeho požadavky k prostředkům např. alokaci šířky pásma a další konfigurační možností pro další specifické koncové zařízení. Před použitím musí být koncové zařízení hostitelem nakonfigurováno. [6]

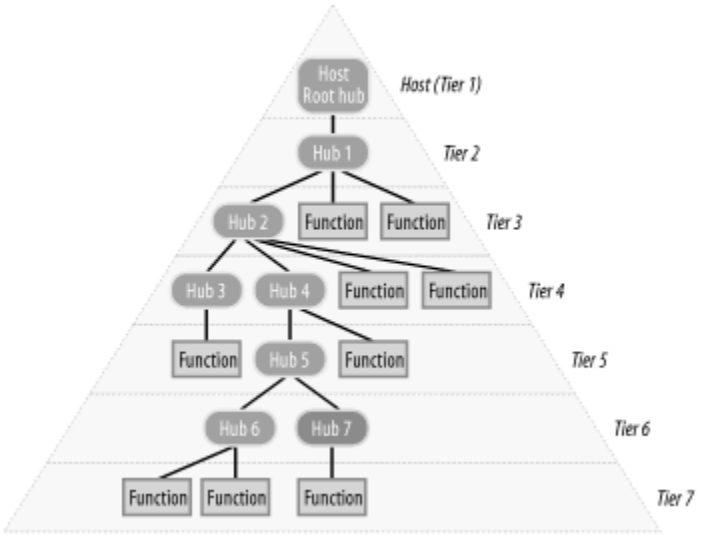
Příklady koncových zařízení můžou být [6]:

- HID jako je myš, klávesnice, gamepad
- Zobrazovací zařízení jako je skener, tiskárna, fotoaparát
- Velkokapacitní zařízení jako je DVD, flash disk

1.3 Fyzická topologie směrnice

Topologie USB je víceúrovňová hvězda. Rozbočovač (hub) je centrem každé hvězdy a každý její článek je dvoubodový spoj mezi hostitelem a rozbočovačem (nebo koncovým zařízením) nebo rozbočovač připojený na jiný rozbočovač (nebo na jiné koncové zařízení). [6]

Pro garantovanou rychlost dat je maximální počet úrovní 7 a to včetně nejvyšší kořenové úrovně, kde je kořenový rozbočovač. V poslední 7. úrovni už není možné mít rozbočovač, ale pouze koncové zařízení. Z toho důvodu je možné mít pouze 5 rozbočovačů připojených v řadě za sebou. Maximální počet připojených zařízení (rozbočovačů a koncových zařízení) na 1 kořenový rozbočovač je 127. [5]



Obr. 2. Fyzická topologie USB [7]

1.4 Protokol sběrnice

USB 2.0 je dotazovací (polled) sběrnice tj. všechny datové přenosy zahajuje pouze hostitelský řadič. Většina transakcí obsahuje přenos až tří paketů. Každá transakce začíná token paketem popisující typ a směr transakce, adresu zařízení a číslo koncového bodu. V dané transakci jsou data přenášena buď z hostitele do zařízení, nebo ze zařízení do hostitele v závislosti na směru přenosu dat. Zdroj transakce poté odešle datový paket nebo oznámí, že už nemá k přenosu žádná data. Úspěšný přenos oznámí cíl hostiteli handshake paketem. [6]

Logický datový přenos používaný mezi hostitelem a koncovým bodem na zařízení je nazýván kanál (pipe). Ten se dělí na stream kanál a message kanál. Liší se v tom, že zatímco message kanály mají definovanou strukturu, data v stream kanálech nemají žadnou. Kanály

obecně vznikají, až když je USB zařízení nakonfigurované a hostitel má požadavek pro přenos. Po dokončení přenosu kanál zanikne. Jakmile je jednou napájené zařízení, bude vždy existovat jeden message kanál tzv. Výchozí řídicí kanál (Default Control Pipe), aby byl zajištěn přístup ke konfiguraci, stavu a informaci o ovládání. [6]

Plánovač transakcí umožňuje řízení datového toku pro některé stream kanály. Na hardwarové úrovni to zabraňuje podtečení nebo přetečení bufferů použitím NAK handshake paketu k omezení rychlosti přenosu dat. Pokud se NAK paket použije, transakce se zopakuje, až se sběrnice uvolní. Mechanismus řízení datového toku (Flow control) umožňuje flexibilní plánování, které přizpůsobuje současně probíhající obsluhy různých stream kanalů. Takhle lze obsluhovat více stream kanalů najednou v různých intervalech a paketů různých velikostí. [6]

Některé transakce mezi hostitelským řadičem a rozbočovači obsahují čtyři pakety. Tyto typy transakcí se používají k řízení datových přenosů mezi hostitelem a full/low-speed zařízeními. [6]

USB 3.2 používá asynchronní notifikace. Zařízení je schopno si asynchronně vyžádat přenos od hostitele. Transakce je iniciována hostitelem vytvořením požadavku, a pokud je zařízení schopno splnit požadavek, data přijme nebo odešle. Např. pokud zařízení není schopno splnit požadavek z důvodu plného bufferu na koncovém bodu, odpoví NRDY (Not Ready) paketem. Až buffer bude volný, zařízení odešle hostiteli ERDY (Endpoint Ready) paket pro znovu odeslání dat bez nutnosti dotazování od hostitele. Pokud je koncový bod zastaven (halted), zařízení odpoví STALL paketem. [8][9]

Hostitel v USB 2.0 vysílá své pakety jako broadcast všem downstream zařízením, USB 3.2 používá unicast vysílání a posílá tedy pakety pouze do cílového zařízení. [9]

USB4 používá protokol založený na specifikacích Thunderbolt 3. Využívá techniku tunelování (tunneling), která umožní používat paralelně více protokolů s pomocí jednotného fyzického rozhraní. [10]

1.5 Robustnost

K robustnosti přispívají tyto vlastnosti [6][9]:

- Integrita signálu pomocí diferenciálních ovladačů, přijímačů a stínění na fyzické úrovni
- CRC ochrana kontrolních a datových polí

- Detekce připojení a odpojení a konfigurace prostředků na úrovni systému
- Vlastní obnovení (Self-recovery) v protokolu používáním časových limitů pro ztracené nebo poškozené pakety
- Řízení datového toku (Flow control) pro přenos streamovaní dat a správu hardwarového bufferu
- Konstrukce datových a řídicích kanálů pro zajištění nezávislosti od nežádoucích interakcí mezi koncovými zařízeními

1.5.1 Detekce a zpracování chyb

Každý paket obsahuje ochranná pole proti chybám. Protokol obsahuje oddělená CRC pro kontrolu a datová pole v každém paketu. U selhané CRC se považuje paket za poškozený. CRC dávají spolehlivou ochranu proti single- a double-bitovým chybám. [6][8]

Protokol umožňuje se vypořádat s chybami hardwarově nebo softwarově. Hardwarové zpracování chyb obsahuje hlášení a opakování neúspěšných přenosů. Když USB hostitelský řadič při přenosu narazí na chybu třikrát za sebou, tak informuje o poruše klientský software, ten pak může data obnovit. [6][8]

1.6 Systémová konfigurace

USB umožňuje zařízení se kdykoliv připojovat a odpojovat bez potřeby manuální konfigurace. [6]

1.6.1 Rozpoznání sběrnice (Bus enumeration)

Rozpoznání sběrnice zařízení slouží k identifikování a přiřazování unikátních adres zařízením připojených ke sběrnici. Protože USB umožňuje zařízení kdykoliv připojovat a odpojovat, je tak rozpoznání sběrnice neustále trvající aktivita pro systémový software USB. [6]

1.6.2 Charakterizace zařízení

USB zařízení jsou zpřístupněna přes USB adresu, která byla po připojení zařízení přiřazena. Každé zařízení používá jeden nebo více kanálů, které hostitel může použít ke komunikaci se zařízením. USB zařízení musí podporovat speciálně navržený kanál pro koncový bod nula (endpoint 0), do kterého USB řídicí kanál bude připojen. Každé zařízení obsahuje informace o sobě samém a stavu zařízení. [6]

1.6.3 Připojování a odpojování zařízení

Rozbočovače používají stavové bity, díky kterým oznamují na jednom z portů připojení nebo odpojení zařízení. Hostitel se na rozbočovač dotáže k získání těchto bitů a v případě připojení hostitel povolí port a adresuje zařízení skrz výchozí řídicí kanál na výchozí adrese. Hostitel přiřadí zařízení jedinečnou adresu a poté zjistí, zda nově připojené zařízení je rozbočovač nebo koncové zařízení. Používáním přiřazené adresy a koncového bodu nula (endpoint 0) hostitel ukončí činnost řídicího kanálu. Pokud připojené zařízení je rozbočovač a další zařízení jsou připojena k jeho portům, pak postup připojení platí pro každé z připojených zařízení k tomuto rozbočovači. Pokud se jedná o koncové zařízení, tak oznámení o připojení bude vyřízena hostitelských softwarem. Když je zařízení odpojeno z portu rozbočovače, rozbočovač deaktivuje port, čímž upozorní hostitele. Odpojení je pak vyřízeno systémovým softwarem USB. Jestli je odpojené zařízení rozbočovač, pak systémový software odstraní tento rozbočovač a všechna zařízení připojená k tomuto rozbočovači. [6]

2 MECHANICKÉ VLASTNOSTI

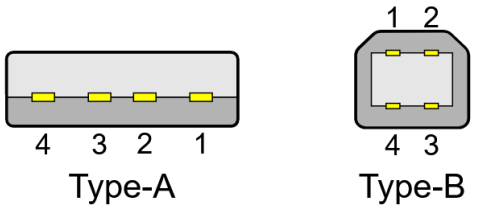
Fyzická topologie spočívá v připojování downstream portu rozbočovače s upstream portem jiného rozbočovače či zařízení. USB 2.0 pracuje s rychlostmi High-speed (480 Mbit/s), full-speed (12 Mbit/s) a low-speed (1,5 Mbit/s). [6]

S příchodem USB 3.2 pracuje USB navíc s rychlostmi 5 Gbit/s u generace 1x1 (Super Speed), 10 Gbit/s u generace 1x2 a 2x1 (Super Speed 10Gbps) a 20 Gbit/s u generace 2x2 (Super Speed 20Gbps). [9]

USB4 pak pracuje s rychlostmi 20 Gbit/s u Gen 2x2 a 40 Gbit/s u Gen 3x2. [11]

2.1 Konektory

Základní konektory USB 1.x a USB 2.0 jsou typy A a B. Typ A připojuje zařízení přímo k hostiteli nebo downstream portu rozbočovače. Typ B se pak připojuje směrem k zařízení. Další možné konektory vycházející z konektorů typu A a B, jsou typy Mini-B, Micro-A, Micro-B. [6]



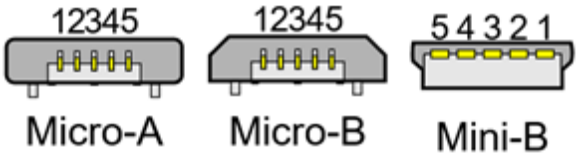
Obr. 3. Schéma konektorů typů A

a B USB 1.x a USB 2.0 (samec) [13]

Tab. 1. Seznam vodičů konektorů typů A a B

USB 1.x a USB 2.0 (samec) [6]

Pin	Jméno signálu	Poznámka
1	V_{BUS}	+5 V
2	D-	Data-
3	D+	Data+
4	GND	Zem

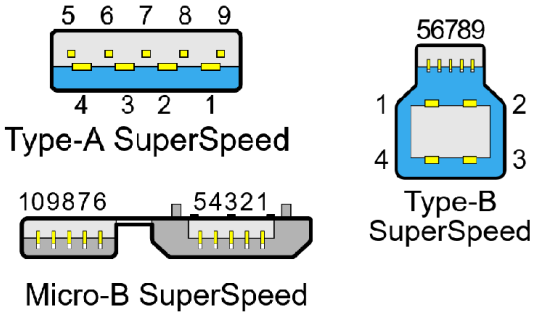


Obr. 4. Schéma konektorů typů Mini-B, Micro-A a Micro-B USB 1.x a USB 2.0 (samec) [14][15][16]

Tab. 2. Seznam vodičů konektorů typů Mini-B, Micro-A a Micro-B USB 1.x a USB 2.0 (samec) [6]

Pin	Jméno signálu	Poznámka
1	V_{BUS}	+5 V
2	D-	Data-
3	D+	Data+
4	ID	On-The-Go ID
5	GND	Zem

USB 3.2 rozlišuje typy konektorů stejně jako předchozí USB 2.0 a jsou konstruovány tak, aby byly zpětně kompatibilní s předchozími generacemi. Mezi stávající typy se přidal pro USB 3.2 Gen 1x2, USB 3.2 Gen 2x2, USB4 Gen 2x2, USB4 Gen 3x2 konektor typ C. [9][11]



Obr. 5. Schéma konektorů typů A, B a Micro-B USB 3.2 (samec) [17][18][19]

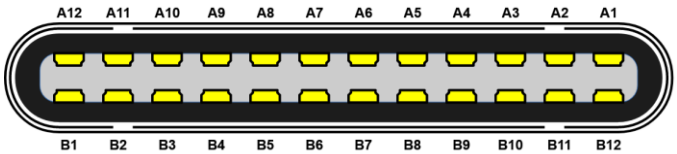
Tab. 3. Seznam vodičů konektorů typů A a B USB 3.2 gen 1x1 a gen 2x1 [9]

Pin	Jméno signálu		Poznámka
	Konektor A	Konektor B	
1	V_{BUS}		Napájení

2	D-		USB 2.0 DATA±
3	D+		
4	GND		Zem
5	StdA_SSRX−	StdB_SSTX−	Přijímač/Vysílač diferenciálního páru USB 3.2
6	StdA_SSRX+	StdB_SSTX+	
7	GND_DRAIN		Zem pro návrat signálu
8	StdB_SSTX−	StdA_SSRX−	Vysílač/Přijímač diferenciálního páru USB 3.2
9	StdB_SSTX+	StdA_SSRX+	

Tab. 4. Seznam vodičů konektoru typu Micro-B USB 3.2 gen 1x1 a gen 2x1 [9]

Pin	Jméno signálu	Poznámka
1	V_{BUS}	Napájení
2	D-	USB 2.0 DATA±
3	D+	
4	ID	On-The-Go ID
5	GND	Zem
6	MicB_SSTX−	Vysílač diferenciálního páru USB 3.2
7	MicB_SSTX+	
8	GND_DRAIN	Zem pro návrat signálu
9	MicB_SSRX−	Přijímač diferenciálního páru USB 3.2
10	MicB_SSRX+	



Obr. 6. Schéma konektoru typu C (samec) [21]

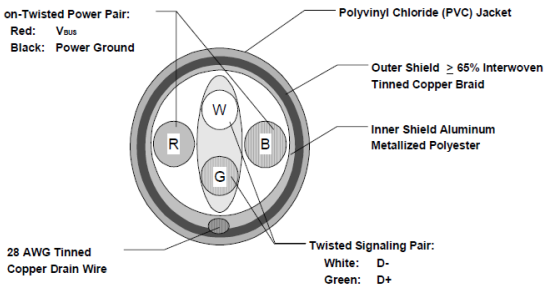
Tab. 5. Seznam vodičů konektoru typu C [9][19]

Pin	Jméno signálu	Poznámka
A1, B1, A12, B12	GND	Zem
A4, B4, A9, B9	V_{BUS}	Napájení

A5	CC	Konfigurační kanál
B5	V_{CONN}	Napájení pro kabel
A6	Dp1	USB 2.0 DATA±
A7	Dn1	
A8	SBU1	Sideband use A
B8	SBU2	Sideband use B
A2	TXp1	Vysílač diferenciálního páru USB 3.2 gen 2x2/USB4
A3	TXn1	
B11	RXp1	Přijímač diferenciálního páru USB 3.2 gen 2x2/USB4
B10	RXn1	
B2	TXp2	Vysílač diferenciálního páru USB 3.2 gen 2x2/USB4
B3	TXn2	
A11	RXp2	Přijímač diferenciálního páru USB 3.2 gen 2x2/USB4
A10	RXn2	

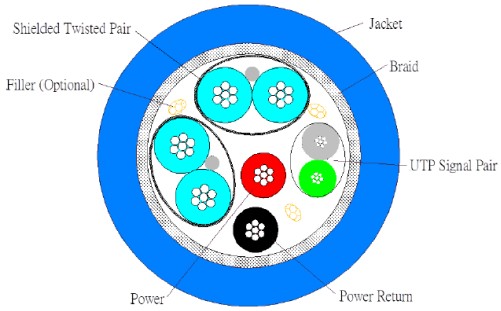
2.2 Kabel

USB 2.0 pro své tři rychlosti používá stíněný kabel se dvěma vodiči (V_{BUS}, GND) pro napájení a kroucenou dvojlinku pro přenos signálů. Existují tři možné varianty kabelů USB. Standard detachable cable je kabel rychlostí high/full-speed, kde na jednom konci je konektor samec typu A a na konci druhém samec typu B. S příchodem konektoru typu Mini-B je možné jím nahradit typ B. High/full-speed captive cable je druh kabele, kde na jednom konci je samec typu A a druhém konci výrobcem specifický konektor pro periferie s používanou rychlostí high/full-speed. Low-speed captive cable je kabel, kde na jednom konci je samec typu A a druhém konci výrobcem specifický konektor pro periferie s používanou rychlostí low-speed. [6]



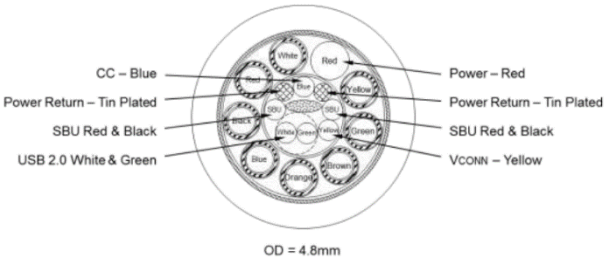
Obr. 7. Průřez kabelu USB 2.0 [6]

USB 3.2 generace 2x1 používá stíněný kabel s dvěma vodiči pro napájení (GND, V_{BUS}), jednu nestíněnou kroucenou dvojlinku (pro přenos dat rychlostí High-speed nebo nižší) a dva stíněné diferenciální signálové páry pro přenos SuperSpeed (stíněná kroucená dvojlinka, twinaxiální kabel). [22]



Obr. 8. Průřez kabelu USB 3.2 gen 2x1
pro konektory A, B, Micro-B [22]

USB-C používá stíněný kabel s dvěma vodiči pro napájení (GND, V_{BUS}), jednu nestíněnou kroucenou dvojlinku (pro High-speed nebo pomalejší), čtyři stíněné diferenciální signálové páry pro přenos SuperSpeed (stíněná kroucená dvojlinka, twinaxiální kabel) a dva sideband signálové vodiče a jeden konfigurační kanál. [19]



Obr. 9. Průřez kabelu pro konektor USB-C [19]

Možné kombinace konektorů v kabelu pro USB 3.2 ukazuje následující tabulka (Tab. 6).

Tab. 6. Kombinace konektorů v kabelu pro USB 3.2 [19]

Plug 1	Plug 2
USB 3.2 Standard-A	USB 3.2 Standard-B
USB 3.2 Standard-A	USB 3.2 Micro-B
USB 3.2 Standard-A	USB 3.2 Standard-A
USB 3.2 Standard-A	USB 3.2 Type-C
USB 3.2 Type-C	USB 3.2 Standard-B
USB 3.2 Type-C	USB 3.2 Micro-B
USB 3.2 Type-C	USB 3.2 Type-C

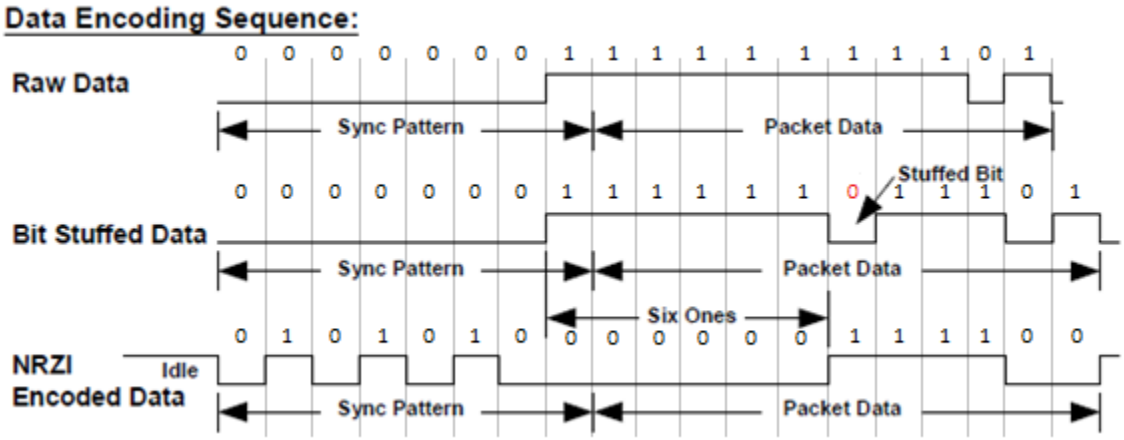
3 FYZICKÉ ROZHRANÍ

3.1 Kódování dat

3.1.1 USB 2.0

Předtím než dojde ke kódování dat, prochází data bit stuffingem pro zajištění synchronizace dat. Po každých šesti následně jdoucích bitových jedniček se vloží jedna nula k vynucení změny stavu signálu. Bit stuffing je umožněn od začátku synchronizačního vzoru (sync pattern). [6][23]

USB 2.0 při přenosu paketů používá pro kódování dat NRZI-S, kde je přiřazena logická nula pro změnu stavu signálu a logická jednička pro jeho zachování stavu signálu. Při dlouhém řetězci nul se stav signálu mění každým poslaným bitem, při řetězci jedniček se stav nemění. Přijímač pak musí tyto data dekódovat a vložené bity odstranit. [6][24]



Obr. 10. Kódování NRZI s bit stuffingem [6]

3.1.2 USB 3.2

USB 3.2 gen 1 pro lepší synchronizaci provádí před kódováním scrambling, což je převod vstupních dat na pseudonáhodnou binární sekvenci definovanou polynomiálem $G(X) = X^{16} + X^5 + X^4 + X^3 + 1$ s pomocí posuvného registru s lineární zpětnou vazbou (linear-feedback shift register). Umožňuje pravidelnější obnovení hodinových signálů (clock recovery) zabráněním dlouhým sekvencím nul nebo jedniček. LFSR je resetován, když je poslán nebo obdržen comma symbol. Poté následuje kódování dat a po dokončení přenosu se data zase dekódují a odstraní scrambling. [8][25][26]

Generace 1 pro kódování používá 8b/10b, ve kterém se 8 bitové řetězce převádějí na 10 bitové symboly. Kódování probíhá tak, že se 8 bitový řetězec rozdělí na dvě části, v první části je prvních 5 bitů (směrem od LSB) a v druhé zbývající 3 bity. Podle kódovacích tabulek se následně u obou částí určí daná sekvence bitů s přidaným jedním bitem a poté se obě části spojí do jednoho řetězce 10 bitů, který se nazývá datovým symbolem. Kódování také definuje 12 speciálních symbolů, které je možné odeslat namísto datového symbolu. Ty slouží k určení činností na linkové vrstvě (např. začátek nebo konec rámce). Pro lepší synchronizaci slouží tzv. comma symboly (označeny jako K.x.y). Některé 8 bitové řetězce lze kódovat dvěma různými způsoby. Aby přenos nul a jedniček byl rovnoměrně rozložen (výhodou je pak větší ochrana proti EMI a lepší opravitelnost chyb), je omezen rozdíl mezi nulami a jedničkami v bitovém řetězci maximálně na ±2, tj. ±1 pro každý ze symbolů. Tento rozdíl se nazývá průběžná disparita (running disparity) a na základě daných pravidel se pak rozhodne, jaký se zvolí způsob kódování. Kódování standardně začíná s průběžnou disparitou -1. [8][9]

Tab. 7. Pravidla pro průběžnou disparitu RD [26]

Předchozí RD	RD vybraného řetězce	Zvolená RD	Další RD
-1	0	0	-1
-1	±2	+2	+1
+1	0	0	+1
+1	±2	-2	-1

Tab. 8. Tabulka kódování 5b/6b [26]

Vstupní řetězec		RD = −1	RD = +1		Vstupní řetězec		RD = −1	RD = +1
	EDCBA	abcdei				EDCBA	abcdei	
D.00	00000	100111	011000		D.16	10000	011011	100100
D.01	00001	011101	100010		D.17	10001	100011	
D.02	00010	101101	010010		D.18	10010	010011	
D.03	00011	110001			D.19	10011	110010	
D.04	00100	110101			D.20	10100	001011	
D.05	00101	101001			D.21	10101	101010	
D.06	00110	011001			D.22	10110	011010	
D.07	00111	111000	000111		D.23	10111	111010	000101
D.08	01000	111001	000110		D.24	11000	110011	001100

D.09	01001	100101	
D.10	01010	010101	
D.11	01011	110100	
D.12	01100	001101	
D.13	01101	101100	
D.14	01110	011100	
D.15	01111	010111	101000

D.25	11001	100110	
D.26	11010	010110	
D.27	11011	110110	001001
D.28	11100	001110	
D.29	11101	101110	010001
D.30	11110	011110	100001
D.31	11111	101011	010100
K.28	11100	001111	110000

Tab. 9. Tabulka kódování 3b/4b [26]

Vstupní řetězec		RD = −1	RD = +1
	HGF	fghj	
D.x.0	000	1011	0100
D.x.1	001	1001	
D.x.2	010	0101	
D.x.3	011	1100	0011
D.x.4	100	1101	0010
D.x.5	101	1010	
D.x.6	110	0110	
D.x.P7	111	1110	0001
D.x.A7	111	0111	1000

Vstupní řetězec		RD = −1	RD = +1
	HGF	fghj	
K.x.0	000	1011	0100
K.x.1	001	0110	1001
K.x.2	010	1010	0101
K.x.3	011	1100	0011
K.x.4	100	1101	0010
K.x.5	101	0101	1010
K.x.6	110	1001	0110
K.x.7	111	0111	1000

USB 3.2 gen 2 před kódováním projde řetězec scramblingem používající polynomiál $G(X) = X^{23} + X^{21} + X^{16} + X^8 + X^5 + X^2 + 1$. [9]

Generace 2 používá 128b/132b, což je variace kódování 64b/66b. Výhoda tohoto kódování je efektivnější přenos dat. Zatímco u 8b/10b je vyžadována ve srovnání s nekódovanými daty o 20% větší šířka pásma, u 128b/132b je vyžadována jen o 3,125% více. Vysílač vkládá před 128 bitový řetězec, který je složen z šestnácti 8 bitových symbolů, 4 bitovou hlavičku. Na rozdíl od kódování 64b/66b, kde hlavička je pouze 2 bitová (01b, 10b), tento formát umožňuje opravit chybu i v hlavičce. Ta obsahuje informaci, jestli řetězec posílá čistá data (0011b) nebo posílá 1 z 15 možných kontrolních řetězců (1100b). Tyto kontrolní řetězce mohou být také částečně zaplněny daty. Hlavička obsahující čtyři nuly nebo čtyři jedničky je považována za chybu. [9]

USB4 Gen 2 používá kódování 64b/66 a USB4 Gen 3 pak 128b/132b. [11]

3.2 Elektrické vlastnosti

Hostitelské řadiče a rozbočovače USB 2.0 mají možnost data přenášená v rychlostech full/low-speed přenášet mezi sebou v rychlosti high-speed. Mezi rozbočovačem a zařízením pak probíhá přenos full/low-speed. Tato možnost minimalizuje dopad full/low-speed zařízení na dostupnou šířku pásmu pro high-speed zařízení. Rychlosti full/low-speed existují pro podporu zařízení s malou šířkou pásma, jako jsou myši, protože širší sběrnici by tato zařízení nevyužila. [6]

Hodinové signály jsou přenášeny a kódovány spolu s diferenciálními daty. Před každým paketem se posílá SYNC pole, které umožňuje přijímači synchronizovat hodinové signály pro další přenosy. [6]

USB 2.0 je poloduplexní sběrnice, zatímco USB 3.2 a USB4 jsou duálně simplexní. Tím umožňuje přenos informací v obou směrech současně. [9][11]

3.3 Napájení

USB zařízení může mít svůj vlastní zdroj napájení (self-powered) nebo je zcela závislé na hostiteli (bus-powered).

Pro komunikaci se zařízení USB v rámci napájení dělí na 3 třídy [6]:

- Nízko výkonová koncová zařízení napájené přes sběrnici (Low-power bus powered functions)
- Vysoko výkonová koncová zařízení napájené přes sběrnici (High-power bus powered functions)
- Koncová zařízení s vlastním napájením (Self-powered functions)

U všech tříd zařízení prochází napětí o nominální hodnotě 5 V, ale hodnoty mohou kolísat v povoleném rozpětí záležící na třídě zařízení a na verzi USB. [6][8][9]

Nízko výkonové koncové zařízení napájené přes sběrnici v USB 2.0 mohou odebírat proud maximálně 100 mA a pracují s napětím 4,4 – 5,5 V. USB 3.2 Gen 1x1, Gen 2x1 můžou odebírat až 150 mA a USB 3.2 Gen 1x2, Gen 2x2 až 250 mA v rozsahu 4,75 – 5,5 V. Všechna koncová zařízení se při připojení chovají jako nízko výkonová a až po konfiguraci hostitelem se umožní odebírat vyšší proudy. [6][8][9]

Vysoko výkonová koncová zařízení napájené přes sběrnici v USB 2.0 mohou odebírat proud až 500 mA a pracují s napětím 4,75 – 5,5 V. USB 3.2 Gen 1x1, Gen 2x1 může odebírat až 900 mA a USB 3.2 Gen 1x2, Gen 2x2 až 1,5 A v rozsahu 4,75 – 5,5 V. [6][8][9]

Koncová zařízení s vlastním napájením odebírají proud ze sběrnice také, ale maximálně jako nízko výkonová. Proud ze sběrnice je potřeba pro detekci a konfiguraci koncového zařízení. Zbývající proud odebírá z vlastního externího zdroje a umožní ho odebírat až do výše vysoko výkonových koncových zařízeních. [6][8][9]

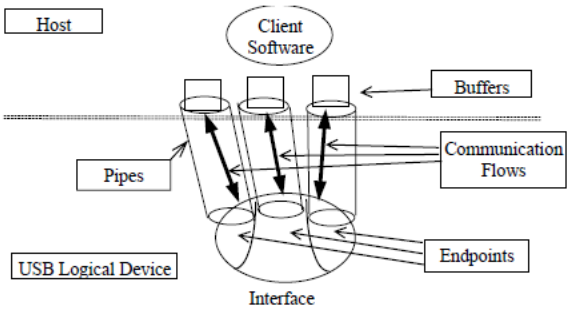
Napájení USB4 je definováno podle specifikací USB-C a používá pro své napájení Power Delivery. Než se ale Power Delivery nakonfiguruje, USB4 pracuje s proudy 250 mA, 1,5 A, nebo 3 A. [19]

3.3.1 Power Delivery

Power Delivery je funkce umožňující dodat větší příkon zařízením. S využitím Power Delivery lze tak získat až 20 V a 5 A. [19]

4 DATOVÁ KOMUNIKACE

Tato kapitola popisuje datovou komunikaci mezi hostitelem a zařízením.



Obr. 11 Datová komunikace [6]

4.1 Koncový bod zařízení (Device Endpoint)

Koncový bod zařízení (endpoint) je jedinečně identifikovatelná část zařízení USB v komunikaci mezi hostitelem a zařízením. Koncový bod je buffer, který ukládá určité množství bytů a tyto data jsou pak buď přijata dál ke čtení, nebo čekají na odeslání, až si data hostitel vyžádá. Název koncového bodu je odvozen od své adresy. Tato adresa se skládá z čísla koncového bodu a směru přenosu. Číslo koncového bodu je hodnota v rozmezí 0-15. Směr je určen z pohledu hostitele, kde vstupní koncové zařízení (IN endpoint) dodává data k odeslání k hostiteli a výstupní koncové zařízené (OUT endpoint) ukládá data přijatá od hostitele. Koncové body lze rozdělit do řídicích koncových bodů (control endpoint) a datových koncových bodů (data endpoint). [6][8]

Řídicí koncové body přenáší data v obou směrech. Skládá se z páru adres koncových bodů IN a OUT, které mají společné číslo koncového bodu. Všechna zařízení po připojení musí poskytnout alespoň jeden řídicí koncový bod na adrese 0 (endpoint 0). Tento koncový bod slouží k rozpoznání a konfiguraci zařízení. [6][8]

Datové koncové body přenáší data v jednom směru, přestože stavové a řídicí informace mohou být přenášena v opačném směru. Jedno číslo koncového bodu může mít zároveň IN a OUT adresu koncového bodu. Full/high-speed zařízení mohou mít až 30 (+ endpoint 0) dalších adres, zatímco low-speed zařízení nejvýše 2 další adresy, ve které mohou být 2 IN, 2 OUT nebo IN a OUT adresy v každém směru. [6][8]

4.2 Kanály (Pipes)

Kanál je spojení mezi koncovým bodem a softwarem hostitelského řadiče, ke kterému dochází na abstraktní úrovni po připojení zařízení. Hostitelský software v průběhu rozpoznávání zařízení vytváří nový kanál s každou adresou koncového bodu, se kterou si hostitel žádá komunikovat. Po vyjmutí zařízení ze sběrnice jsou kanály hostitelem zrušeny. Kanály také mohou být vytvářeny nebo rušeny s pomocí řídicích přenosů (control transfer) podle požadavků hostitele v případě jiné možné konfigurace. Koncový bod 0 má vlastní kanál zvaný Výchozí řídící kanál (Default Control Pipe) a umožňuje přijímat všechny stavové a řídicí požadavky dokud je zařízení stále připojeno. Kanály se dělí na stream kanály a message kanály. Výchozí řídicí kanál je message kanál. [6][8][27]

4.2.1 Stream kanály

Stream kanál nemá definovanou strukturu podle USB specifikací. Stream kanály jsou jednosměrné a data jsou přenášena v sekvenčním pořadí FIFO. Jsou k zařízení vázány k jednomu číslu koncového bodu zařízení ve vybraném směru, tzn., že číslo koncového bodu zařízení pro opačný směr musí použít jiný stream kanál. Stream kanály využívájí přenosy hromadné, izochronní a přenosy přerušení. [6][8]

4.2.2 Message kanály

Message kanál má definovanou USB strukturu. Přenos začíná požadavkem od hostitele do zařízení. Poté následuje přenos stavových informací ve vybraném směru. Tyto požadavky jsou posílány v sekvenčním pořadí FIFO. Message kanál umožňuje přenos oběma směry. Message kanál u zařízení vyžaduje jedno číslo koncového bodu zařízení v obou směrech. USB System software zajišťuje, že se do stream kanálu současně neposílá více požadavků, protože message kanál zvládne obsluhovat pouze jeden message požadavek najednou. Message kanály využívají pouze řídicí přenosy. [6][8]

4.3 Rámce a mikrorámce (Frames)

USB specifikace definuje rámec jako časový úsek, ve kterém probíhají transakce. Pro low/full-speed sběrnici je rámec 1 ms, pro high-speed má mikrorámec délku 125 μs. Do rámce či mikrorámce se může vejít několik transakcí. [6][27]

4.4 Přenos (Transfer) USB 2.0

Každý přenos se skládá z jednoho nebo více transakcí. [6]

4.4.1 Typy transakcí

Každá transakce začíná paketem, který obsahuje číslo koncového bodu, směr přenosu dat (IN, OUT) a jestli zahájí řídicí přenos. Existují 3 druhy transakcí. Dvě z nich jsou pojmenované podle směru koncového bodu (IN transakce, OUT transakce) a třetí Setup transakce zahajuje řídicí přenos (ale chová se jako OUT transakce) a je to jediná transakce, kterou musí zařízení po připojení přijmout. Zbývající přenosy používají IN transakce nebo OUT transakce. Transakce se skládá z dvou nebo tří paketů [6]:

- Token paket
- Datový paket
- Handshake paket (u izochronních přenosů se nepoužívá)

4.4.1.1 Split transakce

Split transakce se používá pouze mezi hostitelským řadičem a rozbočovačem používající rychlost high-speed, pokud je k rozbočovači připojeno zařízení s rychlostí low/full speed. Transakce zabraňuje, aby tato zařízení zpomalily celou sběrnici. [6]

4.4.2 Formát paketových polí

Paket se skládá z různých typů polí. Mohou to být pole SYNC, PID, Adresová, Datová, Endpointová, CRC a EOP. [28]

4.4.2.1 SYNC pole

Synchronizační pole (SYNC) je používáno pro synchronizaci hodinových signálů. Pro low/full-speed je délka pole 8 bitů, pro high-speed je 32 bitů. Přijatá SYNC pole mohou být kratší. Poslední 2 bity označují, kde SYNC pole končí. [28][29]

4.4.2.2 PID pole

První byte po synchronizačním poli je Paket ID (PID), který obsahuje informace o druhu paketu při transakci. Samotný PID je dlouhý 4 bity, ale bity jsou opakovány a slouží pro kontrolu. Pokud je PID poškozený, paket není přijat. [28][29]

Tab. 10. Typy Paket ID [6][27]

Typ paketu	Název PID	Binární zápis
Token	OUT	0001b
	IN	1001b
	SOF	0101b
	SETUP	1101b
Data	DATA0	0011b
	DATA1	1011b
	DATA2	0111b
	MDATA	1111b
Handshake	ACK	0010b
	NAK	1010b
	STALL	1110b
	NYET	0110b
Special	PRE	1100b
	ERR	1100b
	SPLIT	1000b
	PING	0100b
	Reserved	0000b

4.4.2.3 Adresové pole

Adresové pole identifikuje zařízení, se kterým hostitel komunikuje. Délka pole je 7 bitů, což znamená adresaci až 127 různých zařízení. Zařízení, které ještě nemá přidělenou adresu, používá výchozí adresu nazvanou adresa 0. [28]

4.4.2.4 Pole koncového bodu

Pole koncového bodu slouží k identifikaci čísla koncového bodu v zařízení. Délka pole jsou 4 bity, tj. až 15 různých koncových bodů v každém směru (+ koncový bod 0). Směr je zahrnut v PID. [28]

4.4.2.5 Pole čísla rámce (Frame number field)

Slouží k identifikaci rámce/mikrorámce. Toto pole o délce 11 bitů se inkrementuje a po získání maximální hodnoty se vynuluje. Hostitel posílá toto pole v SOF (Start-of-frame) paketu při každém začátku rámce/mikrorámce. [6]

4.4.2.6 Datové pole

Datové pole slouží k přenosu dat, délka pole je proměnlivá 0-1024 bytů. U low-speed zařízení je to až 8 bytů, full-speed až 1023 bytů a high-speed je maximální přenos v jednom poli pak 1024 bytů. [6]

4.4.2.7 Cyklický redundantní součet CRC

CRC slouží pro detekci chyb v paketu. Pokud CRC má při kontrolním součtech různé výsledky, příjemce bude tento paket ignorovat. Token pakety používají 5 bitový CRC a datové pakety 16 bitový CRC. [28]

4.4.2.8 Ukončení paketu (EOP)

Každý paket je ukončen EOP, elektrickým signálem Single Ended Zero (SE0). [28]

4.4.3 Typy paketů

4.4.3.1 Token paket

Skládá se z polí PID, adresových, koncového bodu a CRC5. V token paketu adresová pole a pole koncového bodu identifikují koncový bod, který v případě SETUP a OUT transakcí obdrží datový paket nebo ho pošle v případě IN transakcí. [28]

Tab. 11. Token paket [29]

Pole	SYNC	PID	ADDR	ENDP	CRC5	EOP
Počet bitů	8/32	8	7	4	5	3

4.4.3.2 Datový paket

Datový paket se skládá z PID, datového pole a CRC16. Celkem existují 4 datové pakety: DATA0, DATA1, DATA2 a MDATA. Low/full-speed zařízení mají pouze DATA0 a DATA1. [6]

Tab. 12. Datový paket [29]

Pole	SYNC	PID	DATA	CRC16	EOP
Počet bitů	8/32	8	0-8192	16	3

4.4.3.3 Handshake paket

Handshake paket se používá zjištění stavu transakce. Tyto pakety mohou hostiteli zpět odesílat informace o úspěšném nebo neúspěšném výsledku transakce. Skládá se jenom z PID, který určí druh handskake paketu. [28]

Tab. 13. Handshake paket [29]

Pole	SYNC	PID	EOP
Počet bitů	8/32	8	3

Existují celkem 5 druhů handshake paketů: ACK, NAK, STALL, NYET a ERR. Paket ACK (acknowledge) oznamuje, že transakce proběhla bezchybně. NAK (negative acknowledge) oznamuje, že koncové zařízení není schopná přijmout od hostitele žádná data nebo nemá žádná data pro přenos. STALL může znamenat neplatný řídicí požadavek kanálu příp. jeho selhání nebo zastavení koncového bodu. NYET (not yet) mohou používat jen high-speed zařízení jako část PING protokolu nebo informuje při split transakci, že data byla sice odeslána, ale není je možné přijmout. ERR je pouze pro high-speed rozbočovače, aby hlásily chyby na low/full-speed sběrnici při split transakci. [6][8]

4.4.3.4 SOF Paket

SOF paket je opakovaně posílán hostitelem každou 1 ms ve full-speed a každých 125 μs pro hish-speed zařízení. Skládá se z PID, čísla pole rámce (poskytuje identifikaci rámce/mikrorámce) a CRC5. [6]

Tab. 14. SOF paket [29]

Pole	SYNC	PID	Číslo rámce	CRC5	EOP
Počet bitů	8/32	8	7	5	3

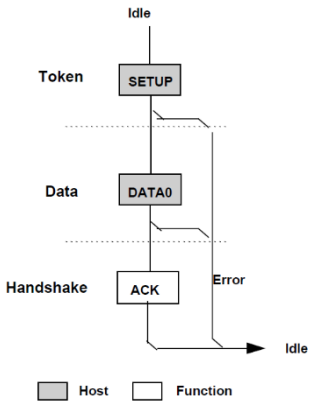
4.4.4 Přenosy

4.4.4.1 Řídicí přenosy

Řídicí přenosy umožňují přístup do zařízení a pro tyto účely používá různé stavové příkazy a konfigurace mezi klientským softwarem a zařízením. Řídicí přenosy používají obousměrný přenos přes message kanály, takže když řídicí kanál je nakonfigurován, používá jak koncový bod IN, tak koncový bod OUT s jedním specifickým číslem koncového bodu. K určení maximální potřebné velikosti paketů musí USB systémový software přečíst

deskriptor zařízení (device descriptor). Na to hostiteli pak stačí přečíst prvních 8 bytů. Řídicí přenos se dělí na tři fáze: setup, data a status. [6][8][28]

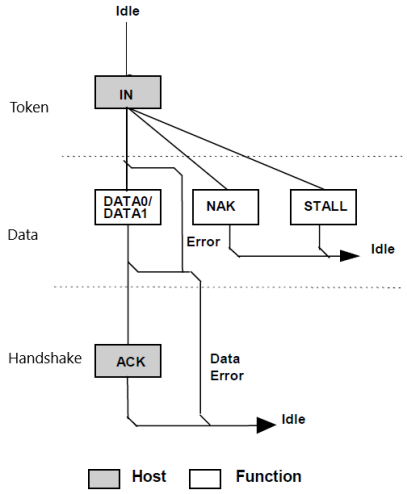
První fáze setup přenáší požadavky od hostitele k funkci. Skládá se ze tří paketů. Jako token paket je zde SETUP paket obsahující adresu a číslo koncového bodu. Jako datový paket je vždy poslán DATA0 paket, který obsahuje informace o daném požadavku. Pokud zařízení obdrží všechna setup data, odpoví ACK handshake paketem. V opačném případě jsou zaslaná data ignorována a k žádné odpovědi nedojde. Délka paketů pro řídicí přenosy je v setup fázi 8 bytů. [6][8][28]



Obr. 12. Řídicí SETUP transakce [6]

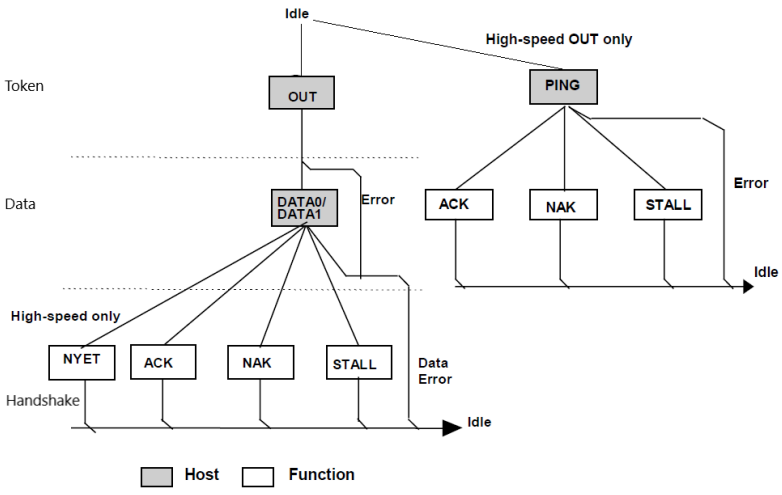
Ve druhé, datové fázi se odesílají data a jsou odesílány ve stejném směru jako v první fázi. Tato fáze je volitelná. Skládá se z jedné (IN nebo OUT) nebo více transakcí a všechny musí být ve stejném směru. Data jsou pak odesílána ve více transakcích, pokud množství dat je překročeno velikostí datového paketu. V datové fázi je maximální délka (tzn., může být i kratší) pro low-speed zařízení 8 bytů, pro full-speed zařízení 8, 16, 32, 64 bytů, pro high-speed je to 64 bytů a pro USB 3.2 je to 512 bytů. Datová fáze je ukončena, když je přeneseno přesné množství dat specifikové v setup fázi nebo je přenesen datový paket o nulové velikosti nebo menší velikosti než je maximální. Můžou nastat dva scénáře v závislosti na směru transakce: IN, OUT. [6][8][9]

IN: K přijetí dat hostitel pošle IN token paket. Jestliže byl paket bezchybně přijat, pak koncové zařízení odpoví datovým paketem a je připraven na další. V případě, že měl nějakou chybu koncový bod, pak zařízení odpoví STALL paketem. V případě funkčního koncového bodu, ale bez žádných dat k odesílání, odpoví NAK handshake paketem. Pokud už má nějakou chybu IN token paket, zařízení bude pakety ignorovat. [6][28]



Obr. 13. Řídicí IN transakce (datová fáze) [6]

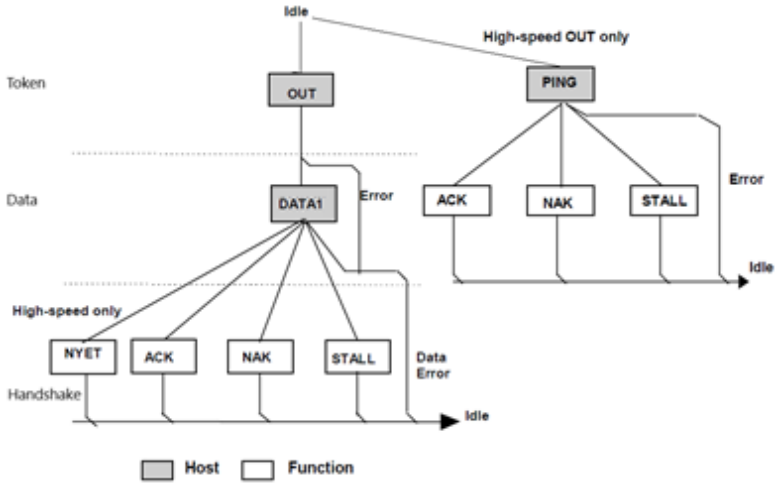
OUT: K odeslání dat hostitel pošle OUT token paket. Pokud dojde k bezchybnému přenosu, zařízení odpoví ACK paketem a je připraven pro další paket nebo odpoví NYET paketem, který také data přijme, ale není připraven přijmout další. NAK paketem odpoví v případě zaneprázdnosti koncového zařízení např. když buffer v koncovém bodu zařízení není prázdný. Pokud by koncový bod měl chybu, koncové zařízení vrátí STALL paket. V případě chybě OUT token paketu nebo datového paketu, bude koncové zařízení pakety ignorovat. Hostitel může ještě před odesláním OUT transakce odeslat PING paket. Odpovědí může být buď ACK paket, pokud je koncové zařízení připraveno k odeslání, NAK paket pokud ještě není připraveno nebo STALL paket pokud koncový bod není schopný z důvodu chyby splnit příkaz. [6][28]



Obr. 14. Řídicí OUT transakce (datová fáze) [6]

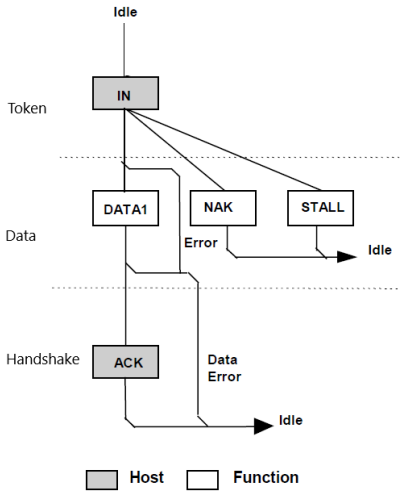
Třetí status fáze vrací hostiteli stavové informace zařízení. Skládá se z jedné IN nebo OUT transakce. Status fáze má opačný směr oproti předchozí datové fázi. Pak následuje paket DATA1 PID, který příjemci dat z datové fáze oznámí stav přenosu. V závislosti na směru transakce mohou nastat dvě varianty IN a OUT, které jsou opačné vůči svým datovým fázím. [6][8][9]

IN: Pokud hostitel v datové fázi odesílá IN token paket, pak v důsledku změny směru v této fázi pošle OUT token paket následovaným DATA1 datovým paketem o nulové velikosti. V případě bezchybné transakce zařízení vrátí ACK paket a je připraven přijímat další příkazy nebo vrátí NYET paket, který také data přijme, ale není připraven přijmout další. Pokud dojde k chybě během zpracování příkazu, zařízení vrátí STALL paket. A pokud zařízení ještě nedokončilo zpracování dat, pak zařízení vrací NAK paket. Hostitel může ještě před odesláním OUT transakce odeslat PING paket podobně jako u datové fáze. [6][28]



Obr. 15. Řídicí OUT transakce (status fáze) [6]

OUT: Pokud hostitel odešle pro odesílání dat v průběhu datové fáze OUT token paket, zařízení v této fázi odešle IN token paket následovaným DATA1 datovým paketem o nulové velikosti. U bezchybného přenosu vrátí ACK paket. Pokud dojde k chybě, koncové zařízení vrátí STALL paket. V případě ještě nezpracovaných dat koncové zařízení vrátí NAK paket. [6][28]

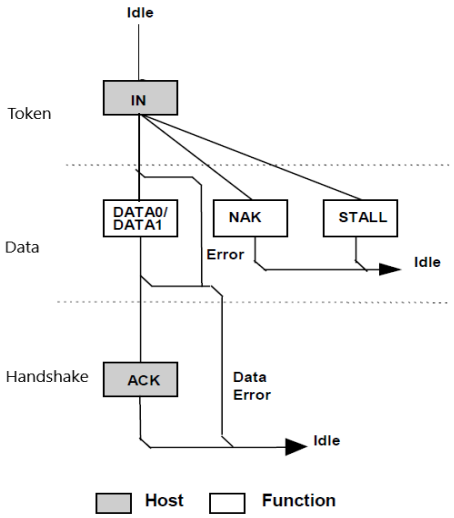


Obr. 16 Řídicí IN transakce (status fáze) [6]

4.4.4.2 Hromadné přenosy

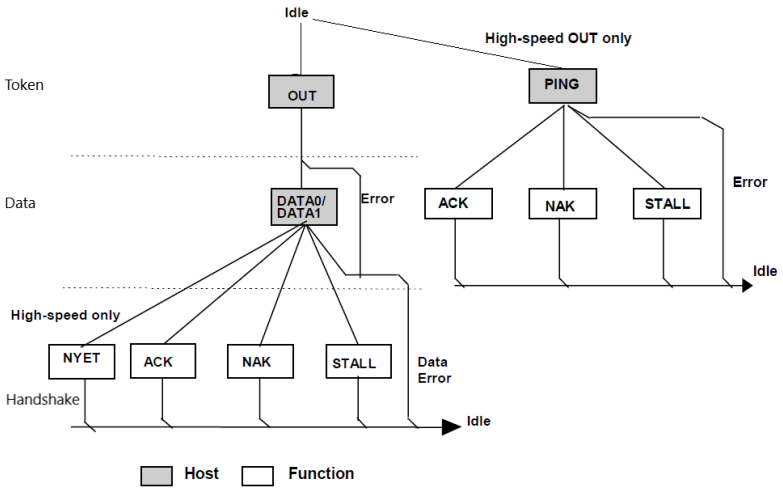
Hromadný přenos je navržen pro přenos velkého množství dat. Tyto přenosy po přidělení šířky pásma jiným přenosům spotřebují veškerou dostupnou šířku sběrnice. K zamezení ucpání sběrnice hromadný přenos svá data pro přenos pozdrží, aby mohly pokračovat jiné typy přenosů. To znamená, že rychlost přenosu je velmi závislá na zatížení sběrnice. Na snížení chybovosti se používá v paketech CRC16 a mechanismy pro znovu zaslání stejných transakcí. Hromadné přenosy používají pro svůj přenos jednosměrné stream kanály, takže je vyžadováno pro přenos v obou směrech dva oddělené kanály a přenosy. Maximální velikost paketu pro full-speed zařízení je 8, 16, 32, 64 bytů, pro high-speed zařízení je to až 512 bytů a pro USB 3.2 je až 1024 bytů. Low-speed zařízení hromadné přenosy nepodporuje. Hromadný přenos je dokončen, pokud je podle požadavku přenesena přesná velikost dat nebo je v datovém paketu množství dat menší než jeho maximální velikost. Pro lepší synchronizaci se u hromadných přenosů používají přepínací bity, ty se přepínají po úspěšném dokončení transakce. [6][8][9]

IN: K přijetí dat hostitel odešle IN token paket. Pokud hostitel získá bezchybná data, odpoví zařízení ACK handshake paketem. V případě, že zařízení nemá dočasně k odeslání žádná data, odpoví NAK paketem. V případě chyby koncového bodu, zařízení odpoví STALL paketem. Pokud zařízení přijme IN token paket s chybou, pak celý paket ignoruje. [6][28]



Obr. 17. Hromadná IN transakce [6]

OUT: K odeslání dat hostitel odešle OUT token paket a po něm následuje datový paket. U bezchybných dat zařízení odpoví hostiteli ACK paketem, který mu oznámí bezchybovost a může pokračovat v přenosu nebo odpoví NYET paketem, který také data přijme, ale není připraven přijmout další. Pokud data byla bez chyby přijata, ale zařízení je nemohlo přijmout z důvodu plného bufferu, odpoví NAK paketem. Pokud došlo k problému v koncovém bodu, zařízení odpoví STALL paketem. Pokud koncové zařízení přijme chybný OUT token paket (nebo datový paket), pak celý paket ignoruje. Hostitel může ještě před odesláním OUT transakce odeslat PING paket. Odpovědí může být buď ACK paket, pokud je koncové zařízení připraveno k odeslání, NAK paket pokud ještě není připraveno nebo STALL paket pokud koncový bod není schopný z důvodu chyby splnit příkaz. [6][28]

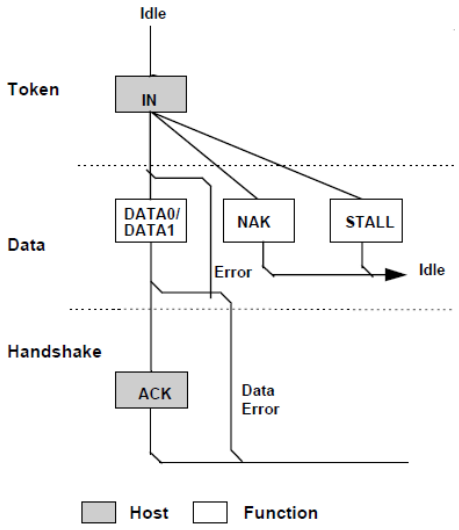


Obr. 18. Hromadná OUT transakce [6]

4.4.4.3 Přenosy přerušení

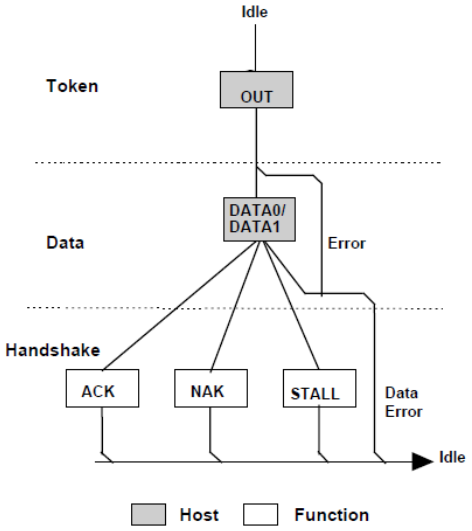
Přenos přerušení se používá pro zařízení s nepravidelně používaným přenosem a rychlou odezvou. Na snížení chybovosti jsou tu mechanismy pro znovu zaslání stejných transakcí. Pro přenos využívá přenos přerušení jednosměrné stream kanály, takže je vyžadováno pro přenos v obou směrech dva oddělené kanály a přenosy. Skládá se z jednoho nebo více (IN, OUT) transakcí. Maximální velikost datového paketu je 8 bytů pro low-speed, 64 bytů pro full-speed a 1024 bytů pro high-speed a USB 3.2 zařízení. Přenos přerušení se dokončí buď přenosem očekávaného množství dat nebo přenosem datového paketu s menším množství dat než je jeho maximální velikost. [6][8][9]

IN: Hostitel pravidelně zařízení posílá dotazy, který obsahuje IN token paket. Pokud se objeví požadavek pro přerušení, koncové zařízení odešle pro přerušení příslušný datový paket. Jestliže koncové zařízení získá datový paket bez problémů, odpoví hostiteli ACK paketem. Pokud data byla poškozená, hostitel bude celý paket ignorovat. Pokud koncový bod nemá žádná nová data k odeslání, koncové zařízení odpoví NAK paketem. Jestliže se objeví chyba na koncovém bodu, koncové zařízení odpoví STALL paketem. Pokud je poškozen IN token paket, koncové zařízení celý paket ignoruje a sleduje sběrnici pro nový IN token paket. [6][28]



Obr. 19. IN transakce přerušení [6]

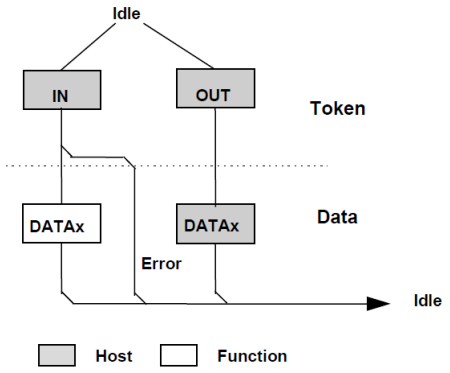
OUT: K odesílání dat hostitel vydá OUT token paket a poté datový paket. V případě prázdného bufferu koncového bodu zařízení a data byla v pořádku, koncové zařízení odešle hostiteli ACK paket. Pokud buffer není prázdný, koncové zařízení odešle NAK paket. Jestli došlo k chybě na koncovém bodu, koncové zařízení odpoví STALL paketem. Pokud je OUT token paket poškozen, pak koncové zařízení ignoruje celý paket. [6][28]



Obr. 20. OUT transakce přerušení [6]

4.4.4.4 Izochronní přenosy

Izochronní přenosy jsou přenosy používající stálý přísun dat stejně velkou rychlostí s možnými občasnými chybami. V případě dat spotřebovávaných stejnou rychlostí (např. video) není sice izochronní přenos vždy vyžadován, ale umí zajistit volnou část sběrnice pro svá data. Izochronní přenos používá jednosměrný stream kanál, takže je vyžadováno pro přenos v obou směrech dva oddělené kanály a přenosy. Přenos se skládá z jedné nebo více (IN, OUT) transakcí o stejných intervalech. Maximální velikost paketu je pro full-speed zařízení 1023 bytů a 1024 bytů pro high-speed i USB 3.2 zařízení. Izochronní přenosy nemají handshake paket, a protože hostitel nemá možnost zjistit chyby v přenosech, pokračuje dál v přenosu, dokud má k dispozici nějaká data. Důvodem je předpoklad, že pro uživatele je méně zaznamenatelné, pokud je chybný paket odeslán než jeho opakování přenosu. Časté chyby v přenosu mohou způsobit paketu změnu velikosti. [6][8][9]



Obr. 21. Izochronní IN, OUT transakce [6]

4.5 Rozdíly v datové komunikaci v USB 3.2

USB 2.0 dělí transakce striktně na token, data a handshake pakety, USB 3.2 u OUT transakce má token paket začleněn do datového paketu a samostatný token paket už není vyžadován. U IN transakce je token paket nahrazen hanshake paketem. Ten je posílán do zařízení k zažadání dat. Zařízení může odpovědět buď vrácením dat, STALL paketem nebo NRDY paketem v případě odložení přenosu dokud zařízení není připraveno. [9]

Hromadné přenosy USB 3.2 mají rozšíření nazvaný streaming, který poskytuje multiplexování nezávislých dat do jednoho koncového bodu. [9]

USB 3.2 umožňuje také bursting, což je přenos bloku paketů, který je odeslán dřív, než je potvrzení přijato příjemcem. [9]

II. PRAKTICKÁ ČÁST

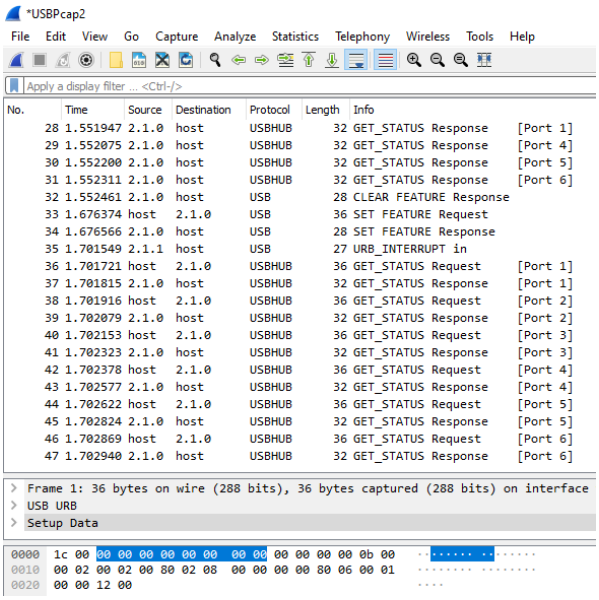
5 SOFTWAROVÉ SNIFFERY

Tato kapitola se zabývá porovnáním softwarovým snifferům.

Softwarové sniffery slouží k zachycení, nahrání a analyzovaní USB komunikace. Tyto sniffery zachytávají URB (USB Request Block) obsaženy v IRP (I/O Request packet). [30]

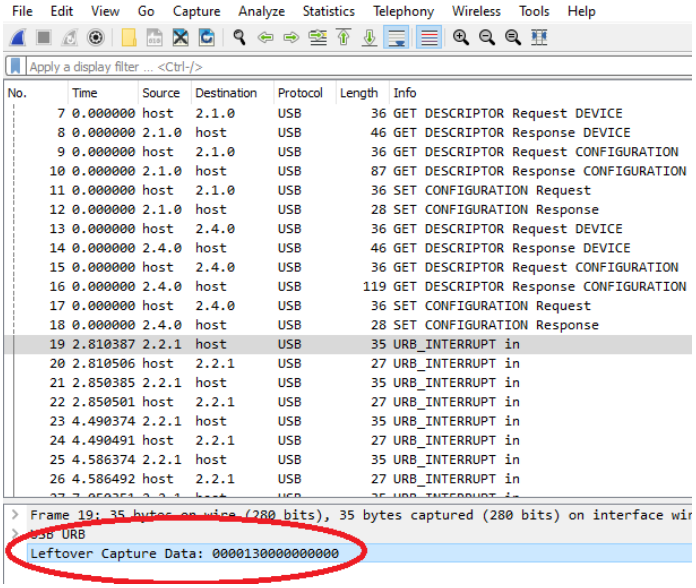
5.1 USBPcap v prostředí Wireshark

USBPcap je open-source sniffer vytvořený Tomaszem Mońem. Veškerá zachycená komunikace je zobrazena v hlavním okně. Po vybrání libovolného paketu lze zjistit jeho podrobné informace. Zvládne zachytit všechny typy přenosů po jednotlivých paketech. Je jediný z testovaných snifferů, který je zdarma. [31]



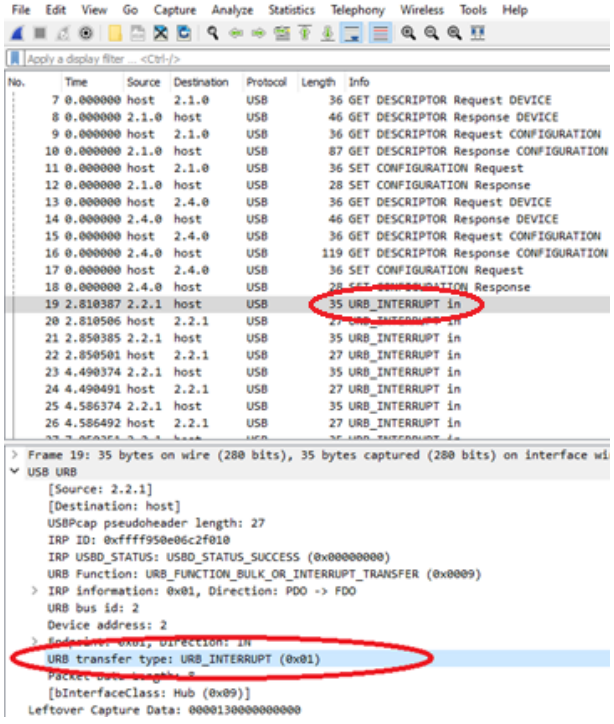
Obr. 22. Základní okno programu Wireshark v průběhu zachytávání komunikace, Zdroj: vlastní zpracování

Obrázek (Obr. 23) popisuje zachytávání komunikace klávesnice v programu Wireshark. Po vybrání paketu s přenosem přerušení lze zachycená data obsahující stisk klávesy najít v položce Leftover Capture Data nebo HID data. S použitím tabulky ve specifikaci USB HID Usage Tables [32] pak lze zjistit o jakou klávesu se jedná.



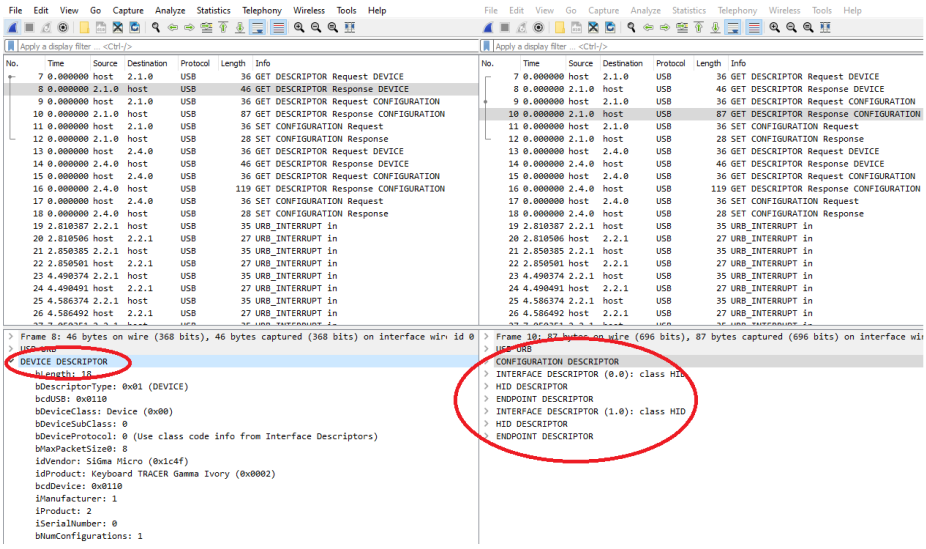
Obr. 23. Zachycená komunikace klávesy v programu Wireshark, Zdroj: vlastní zpracování

Obrázek (Obr. 24) popisuje, kde lze najít v paketu typ datového přenosu v programu Wireshark. Po vybrání paketu lze zjistit typ přenosu buď v položce USB URB pod názvem URB transfer type. Typ přenosu je možné zjistit i přímo na kartě Info (některé pakety zde mají napsaný přesnější popis funkce paketu, např. požadavek o konfigurační deskriptor).



Obr. 24. Zjištění typu přenosu v programu Wireshark, Zdroj: vlastní zpracování

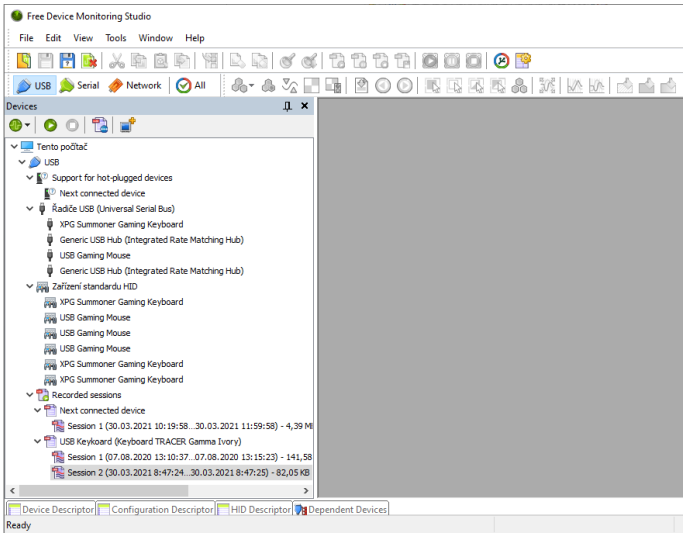
Obrázek (Obr. 25) popisuje v programu Wireshark umístění deskriptorů. Ty jsou ve dvou různých paketech. V paketu GET DESCRIPTOR Response DEVICE je umístěn deskriptor zařízení, v paketu GET DESCRIPTOR Response CONFIGURATION je umístěn konfigurační deskriptor s dalšími deskriptory.



Obr. 25. Deskriptory v programu Wireshark, Zdroj: vlastní zpracování

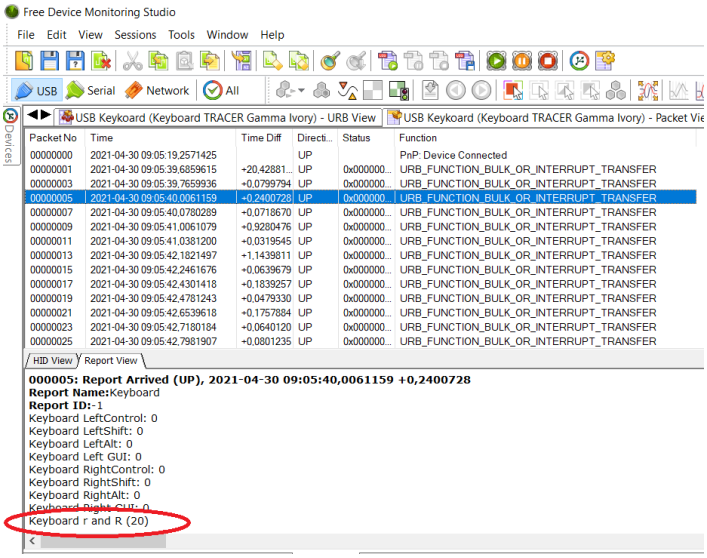
5.2 Device monitoring studio

Sniffer od společnosti HHD Software Ltd. Kromě standardního zachycení, má různé filtry (tzv. View)pro usnadnění analýzy paketů a po okamžitém spustění programu má přístup k informacím z deskriptorů zařízení, řídicí přenosy nezachytává. Umožňuje zachytávat pouze jedno konkrétní zařízení. Device monitoring studio je placený software. [33]



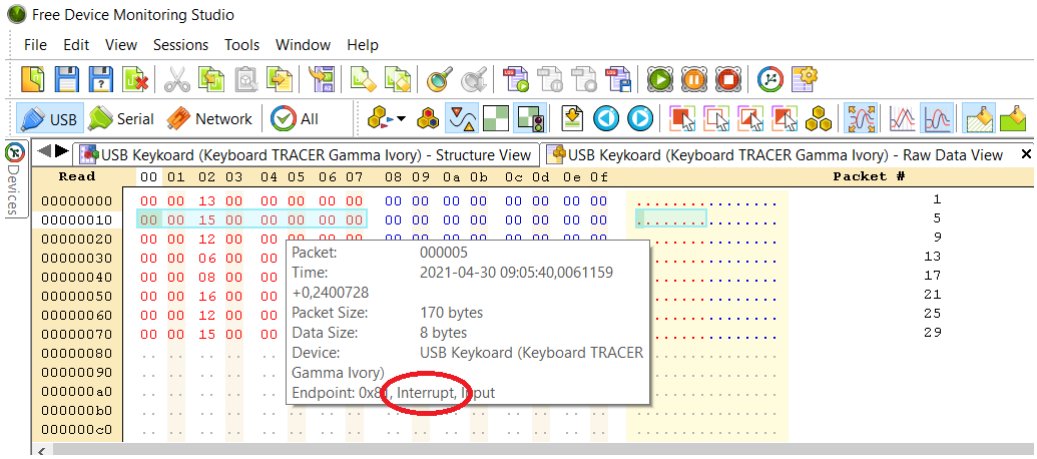
Obr. 26. Základní okno programu Device Monitoring Studio, Zdroj: vlastní zpracování

Obrázek (Obr. 27) popisuje zachycenou komunikaci na klávesnici v programu Device Monitoring Studio. Při výběru možnosti zachytávání z klávesnice je odfiltrována veškerá komunikace z jiných zařízení. Pokud je vybrán pro vizualizaci filtr Packet View, pak po vybrání paketu lze zjistit v kartě Report View přímo stisknutou klávesu, kterou je jinak potřeba najít v tabulkách ve specifikaci USB HID Usage tables. Pokud je potřeba stisk klávesy uvést ve své hexadecimální podobě, pak lze využít filtr URB View nebo Raw Data View.



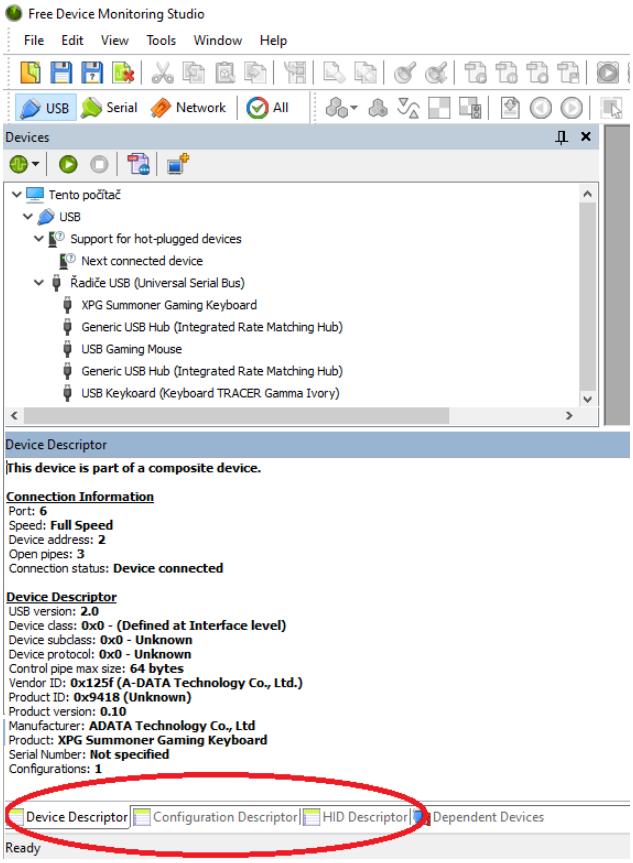
Obr. 27. Zachycená komunikace klávesy v programu Device Monitoring Studio, Zdroj: vlastní zpracování

Obrázek (Obr. 28) popisuje zjištění typu přenosu v programu Device Monitoring Studio. To lze zjistit při výběru filtru Raw Data View a zajet kurzorem myši na vybraný paket.



Obr. 28. Umístění typu přenosu v programu Device Monitoring Studio, Zdroj: vlastní zpracování

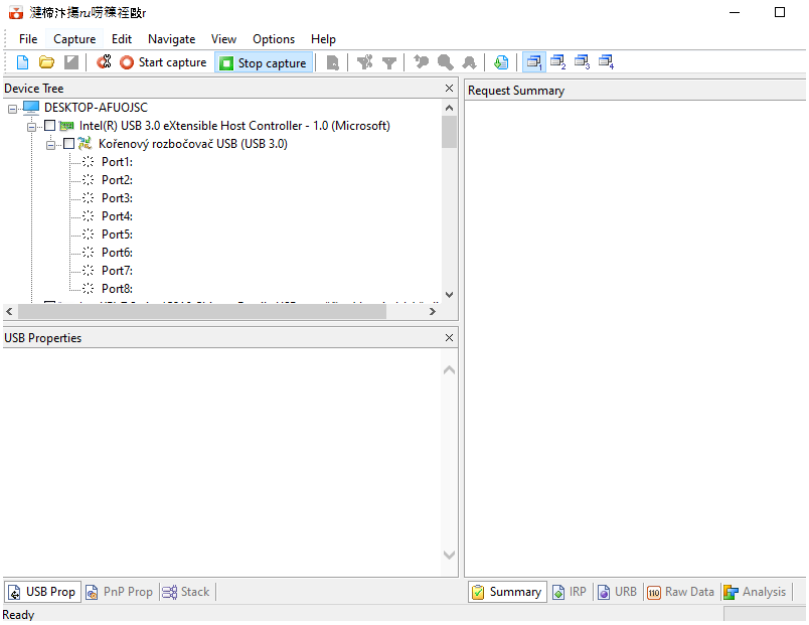
Obrázek (Obr. 29) popisuje umístění karet o deskriptorech v zařízení. Bez potřeby spustění zachytávání komunikace stačí vybrat zařízení, u kterého je potřeba zjistit bližší informace a vybrat daný deskriptor.



Obr. 29. Deskriptory v programu Device Monitoring Studio, Zdroj: vlastní zpracování

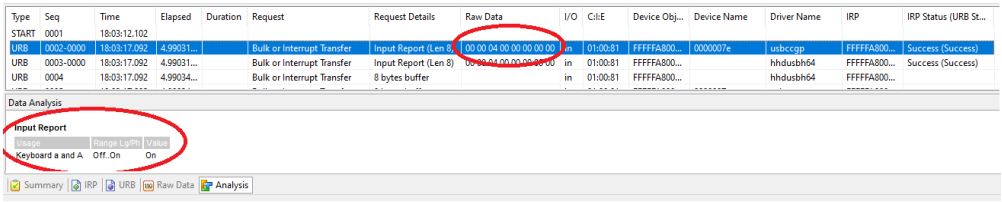
5.3 USBlyzer

Sniffer od společnosti USBlyzer Team. Umožňuje zachytávat komunikaci ve všech kořenových rozbočovačích současně nebo i jedno konkrétní zařízení. Po spuštění má přístup k informacím z deskriptorů zařízení. Nemá možnost zachytávat řídicí přenosy. USBlyzer je placený software. [34]



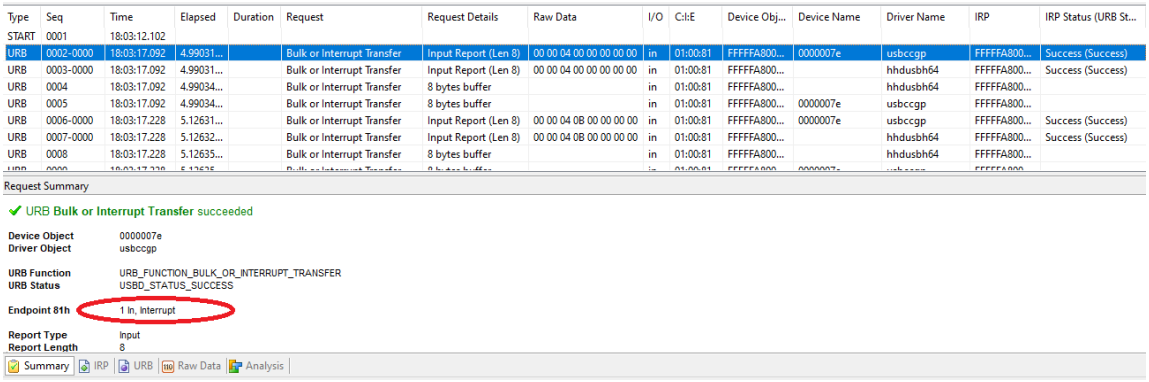
Obr. 30. Základní okno programu USBlyzer, Zdroj: vlastní zpracování

Obrázek (Obr. 31) popisuje zachycenou komunikaci klávesnice v programu USBlyzer. Po výběru klávesnice lze potřebná data pro zjištění klávesy zahlédnout ve sloupci Raw Data v každém paketu, který jej obsahuje. Po stisku určitého paketu je možné zjistit další informace o paketu. V dolním panelu po stisku Analysis je možné zjistit konkrétně, jaká klávesa byla stisknuta bez potřeby ji vyhledat v tabulkách ve specifikaci USB HID usage table.



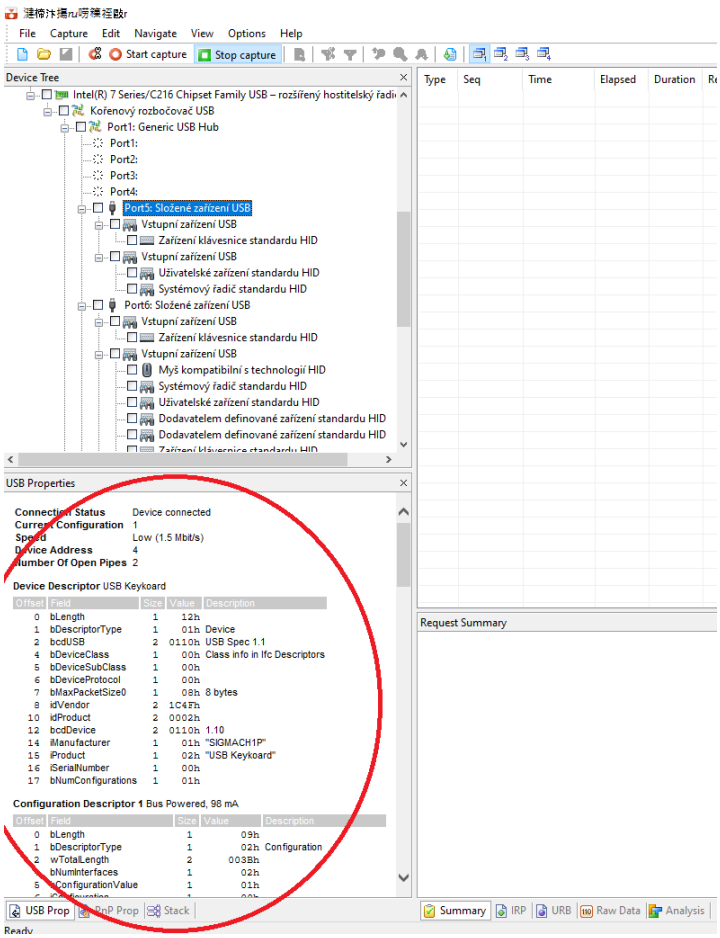
Obr. 31. Zachycená komunikace klávesy v programu USBlyzer, Zdroj: vlastní zpracování

Obrázek (Obr. 32) popisuje zjištění typu přenosu v programu USBlyzer. Po výběru určitého rozbočovače nebo zařízení pro zachytávání komunikace, stačí vybrat daný paket a v dolním panelu vybrat tlačítko Summary.



Obr. 32. Zjištění typu přenosu v programu USBlyzer, Zdroj: vlastní zpracování

Obrázek (Obr. 33) popisuje umístění informací o deskriptorech v programu USBlyzer. Podobně jako Device Monitoring Studio není potřeba spustit zachytávání a lze zjistit tyto informace stiskem na dané zařízení, kde se pak objeví níže v okně USB Prop.



Obr. 33. Deskriptory v programu USBLyzer, Zdroj: vlastní zpracování

5.4 Shrnutí

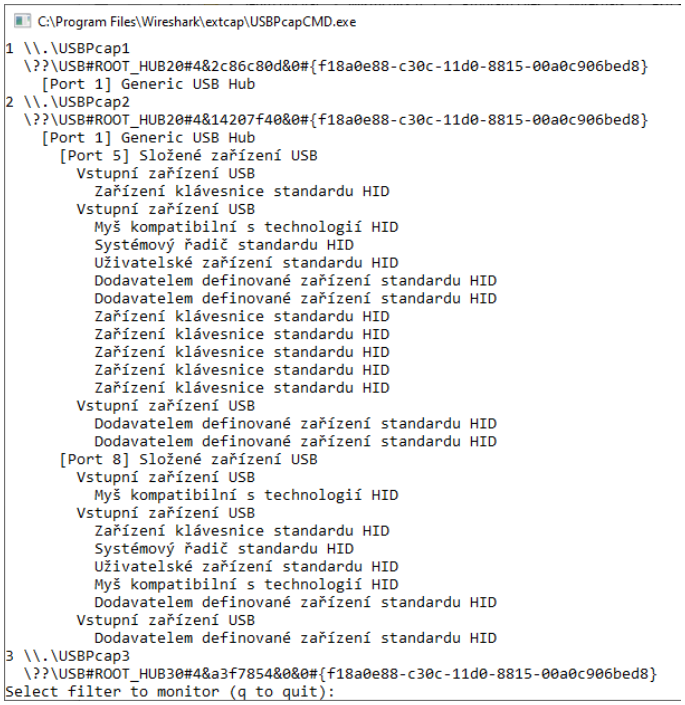
USBlyzer je schopný libovolně zachytávat od konkrétního zařízení až po všechny kořenové rozbočovače. USBPcap je schopný zachytávat kořenové rozbočovače bez možnosti zachytávat samotná jednotlivá zařízení. Device Monitoring Studio zachytává pouze konkrétní zařízení. Device Monitoring Studio a USBlyzer jsou schopny zachytávat přenosy přerušení, hromadné a izochronní. USBPcap je schopen zachytávat všechny typy přenosů. Všechny testované sniffery jsou schopny zachytávat HID zařízení a informace o deskriptorech. USBPcap je schopný zachytit text v textovém editoru při kopírování na flash disk. Z testovaných snifferů pro možné využití v řešení úloh vychází nejlépe program USBPcap v prostředí Wireshark.

Tab. 15. Porovnání možností snifferů

	USBPcap	DMS	USBlyzer
Limity Zachytávání zařízení	Jeden kořenový rozbočovač	Jedno konkrétní zařízení	Všechny kořenové rozbočovače i konkrétní zařízení
Zachytávání přenosů	Všechny typy	Přenosy přerušení, hromadné, izochronní	Přenosy přerušení, hromadné, izochronní
Informace o deskriptorech	Ano	Ano	Ano
Zachytávání textu při přenosu	Ano	Ne	Ne

6 SEZNÁMENÍ S USBPCAP V PROSTŘEDÍ WIRESHARK

Nejprve je třeba určit kořenový rozbočovač, jehož komunikace se bude zachytávat a to podle toho, k jakému kořenovému rozbočovači je zařízení připojené. To lze zjistit např. spuštěním samotné aplikace USBPcap bez programu Wireshark. [35]



Obr. 34. USBPcap, Zdroj: vlastní zpracování

V programu Wireshark se zvolí kořenový rozbočovač, u kterého se zachytává komunikace. Zachytávání komunikace probíhá automaticky a trvá do té doby, dokud ji uživatel nezastaví nebo nenastaví určitý počet zachycených paketů. [36]

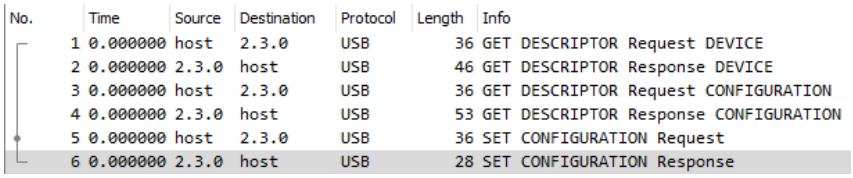
Na hlavním panelu nástrojů jsou ikony dalších funkcí. Např. ruční start/zastavení/restart zachytávání, ukládání do souboru atd. [36]

Obr. 35. Panel nástrojů, Zdroj: vlastní zpracování

Pokud v průběhu zachytání dojde ke komunikaci USB, pak v tabulce paketů lze vidět zachycené informace v jednotlivých sloupcích a jednotlivé řádky představují jeden paket. Jsou to tyto zachycené informace [36]:

- No. – číslo zachyceného paketu (počítá se od startu zachytávání), také označuje za pomocí linky pakety, které spolu nějak souvisí (např. odpověď na požadavek)
- Time – doba zachycení paketu (počítá se od startu zachytávání)

- Source – odesílatel komunikace
- Destination – příjemce komunikace
- Protocol – název protokolu/sběrnice používané při komunikaci
- Lenght – délka paketu
- Info – základní popis paketu



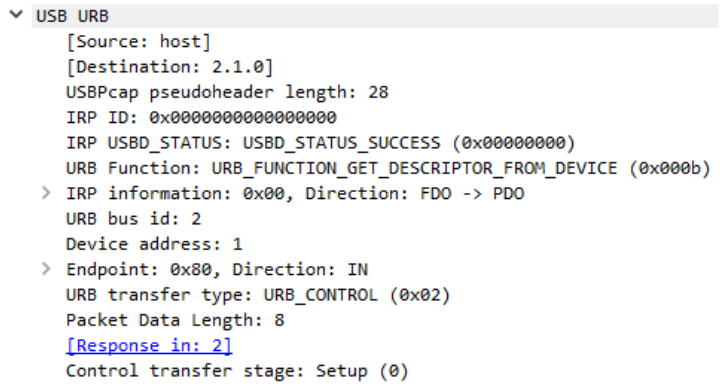
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	host	2.3.0	USB	36	GET DESCRIPTOR Request DEVICE
2	0.000000	2.3.0	host	USB	46	GET DESCRIPTOR Response DEVICE
3	0.000000	host	2.3.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
4	0.000000	2.3.0	host	USB	53	GET DESCRIPTOR Response CONFIGURATION
5	0.000000	host	2.3.0	USB	36	SET CONFIGURATION Request
6	0.000000	2.3.0	host	USB	28	SET CONFIGURATION Response

Obr. 36. Tabulka paketů, Zdroj: vlastní zpracování

Kliknutím na libovolný paket (v programu Wireshark označený jako Frame) lze prozkoumat jeho strukturu a to buď v tabulce podrobností, nebo v tabulce pro hexadecimální výpis. Každý paket má v tabulce podrobností položky Frame a USB URB. Položka Frame obsahuje rozšířené informace (tzv. pole) o paketu v době zachytávání (např. čas přenosu samotného paketu) vygenerované programem Wireshark a tedy nejsou obsaženy v samotném paketu. Informace takto vygenerované jsou označeny hranatými závorkami a lze najít i v položce USB URB. USB URB (USB Request Block) obsahuje podrobné informace o požadavcích a stavu paketu. Jsou to [30]:

- [Source] – název odesílatele
- [Destination] – název příjemce
- USBPcap pseudoheader length – délka hlavičky paketu v bytech
- IRP ID – obsahuje ID I/O Request paketu
- IRP USBD_STATUS – stavový kód pro USB požadavky
- URB Function – specifikuje požadavek pro URB
- IRP information, Direction – určuje směr I/O request paketu
- URB bus id – ID kořenového rozbočovače, na kterém je připojené zařízení
- Device address – adresa zařízení
- Endpoint, Direction – číslo koncového bodu a směru přenosu
- URB transfer type – typ datového přenosu

- Packet Data Length – celková délka dat přenesená v paketu
- [Request/Response in] – odkaz na požadavek/odpověď paketu
- [Time from request] – uběhlá doba odpovědi na požadavek (pouze u paketů poslané hostiteli)
- Control transfer stage – fáze řídicího přenosu (pouze u řídicích přenosů)



Obr. 37. Struktura USB Request blocku, Zdroj: vlastní zpracování

Pakety mohou dále obsahovat i další položky, které jsou volitelné a záleží na funkci paketu.

6.1 Setup data

Obsahují informace a požadavky o konfiguraci zařízení [6]:

- bmRequestType – určuje směr, typ a příjemce požadavku
- bRequest – obsahuje konkrétní požadavek v paketu
- Descriptor Index – index deskriptoru
- bDescriptorType – typ deskriptoru
- Language Id – ID podporovaného jazyka
- bConfigurationValue – hodnota zvolené konfigurace
- wIndex – index umožňující předání parametrů požadavky
- wLength – obsahuje počet Bytů určené k přenosu

6.2 Leftover Capture Data/HID Data

Tato položka obsahuje data pro přenos. Používá se u přenosů přerušení. [37]

6.3 Deskriptory

6.3.1 Deskriptor zařízení

V deskriptoru zařízení jsou tyto pole [6]:

- bLength – velikost deskriptoru v Bytech
- bDescriptorType – typ deskriptoru
- bcdUSB – představuje nejvyšší podporovanou verzi USB v zařízení
- bDeviceClass – kód třídy (používá operační systém k zjištění ovladačů třídy)
- bDeviceSubClass – kód podtřídy (používá operační systém k zjištění ovladačů třídy)
- bDeviceProtocol – kód protokolu (používá operační systém k zjištění ovladačů třídy)
- bMaxPacketSize0 – maximální velikost paketu pro koncový bod nula
- idVendor – ID výrobce zařízení
- idProduct – ID produktu zařízení
- bcdDevice – určuje čas vydání začízení
- iManufacturer – index deskriptoru řetězce výrobce (není povinné, pak je tu nula)
- iProduct - index deskriptoru řetězce produktu (není povinné, pak je tu nula)
- iSerialNumber - index deskriptoru řetězce sériového čísla (není povinné, pak je tu nula)
- bNumConfigurations – ukazuje počet možných konfigurací na zařízení

6.3.2 Konfigurační deskriptor

V konfiguračním deskriptoru jsou tyto pole [6]:

- bLength – velikost deskriptoru v Bytech
- bDescriptorType – typ deskriptoru
- wTotalLength – celková délka dat konfiguračního deskriptoru včetně dalších deskriptorů kromě deskriptoru zařízení v Bytech
- bNumInterfaces – popisuje počet rozhraní v této konfiguraci

- bConfigurationValue – hodnota vybrané konfigurace
- iConfiguration – index deskriptoru popisující tuto konfiguraci
- Configuration bmAttributes – specifikuje parametry pro napájení zařízení
- bMaxPower – specifikuje maximální množství el. proudu napájené ze sběrnice

6.3.3 Deskriptor rozhraní

V deskriptoru rozhraní jsou tyto pole [6]:

- bLength - velikost deskriptoru v Bytech
- bDescriptorType – typ deskriptoru
- bInterfaceNumber – číslo deskriptoru rozhraní
- bAlternateSetting – hodnota pro alternativní rozhraní
- bNumEndpoints – počet koncových bodů používaných rozhraním
- bInterfaceClass – kód třídy (používá operační systém k zjištění ovladačů třídy)
- bInterfaceSubClass - kód podtřídy (používá operační systém k zjištění ovladačů třídy)
- bInterfaceProtocol - kód protokolu (používá operační systém k zjištění ovladačů třídy)
- iInterface – index string deskriptoru popisující toto rozhraní

6.3.4 Deskriptor koncového bodu

V deskriptoru koncového bodu jsou tyto pole [6]:

- bLength – velikost deskriptoru v Bytech
- bDescriptorType – typ deskriptoru
- bEndpointAddress – adresa koncového bodu
- bmAttributes – specifikuje typ datového přenosu, případně jeho další atributy
- wMaxPacketSize – maximální velikost dat pro přenos pro tento koncový bod
- bInterval – specifikuje interval dotazování (polling) pro datové přenosy v rámcích

6.3.5 HID deskriptor

V deskriptoru rozhraní jsou tyto pole [32]:

- bLength – velikost deskriptoru v Bytech
- bDescriptorType – typ deskriptoru (HID)
- bcdHID – číslo vydání USB specifikací pro rozhraní
- bCountryCode – kód země lokalizovaného zařízení
- bNumDescriptors – počet HID třídních deskriptorů
- bDescriptorType – specifikace typu deskriptoru (volitelné, obvykle report)
- wDescriptorLength – celková délka volitelného deskriptoru v Bytech

6.3.6 Deskriptor řetězce

V deskriptoru řetězce jsou tyto pole [6]:

- bLength – velikost deskriptoru v Bytech
- bDescriptorType – typ deskriptoru
- wLANGID – kód pro podporovaný jazyk

Všechny následné deskriptory řetězců vypadají takto [6]:

- bLength - velikost deskriptoru v Bytech
- bDescriptorType - typ deskriptoru
- bString – ukazatel na buffer obsahující unicode řetězec s požadovaným deskriptorem řetězce

Tabulka pro hexadecimální výpis obsahuje na každém řádku datový offset o 16 hexadecimálních bytech a jeho výpis v ascii.

```
0000  1b 00 10 f0 c2 06 0e 95  ff ff 00 00 00 00 09 00   ........ ........
0010  01 02 00 02 00 81 01 08  00 00 00 00 00 13 00 00   ........ ........
0020  00 00 00                                          ...
```

Obr. 38. Hexadecimální výpis paketu, Zdroj: vlastní zpracování

7 PŘÍKLADY KOMUNIKACE

Vzorové příklady pro komunikaci po USB.

7.1 Příklad 1. Bit stuffing a kódování NRZI-S

Před kódováním nejprve proběhne vkládání bitů tzv. bit stuffing, pak následuje samotné kódování NRZI-S. Ukázka RAW dat:

01011001 10111111 10110011 1111

7.1.1 Bit stuffing

Bit stuffing se přidává kontrolní logickou 0, jestliže jde za sebou 6 logických jedniček.

01011001 10111111 01011001 111110

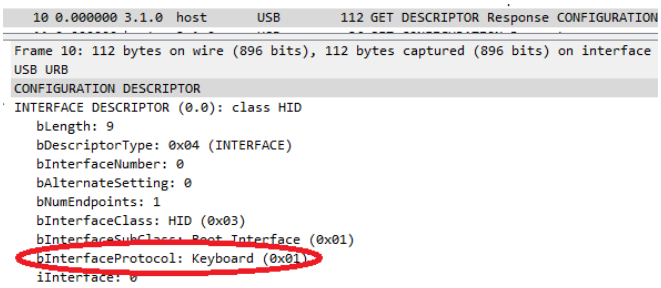
7.1.2 NRZI-S

Logická nula mění stav signálu, logická jednička ji nemění. Grafická ukázka kódování již byla ukázána na obrázku (Obr. 10).

00111011 10000000 11000100 000001

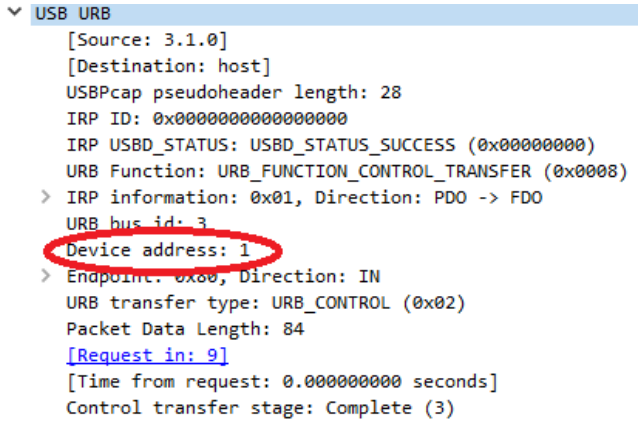
7.2 Příklad 2. Zjištění adresy zařízení (Device address)

Pro příklad byl použit program USBPcap v prostředí Wireshark. Nejprve je třeba zjistit, které pakety komunikují mezi hostitelem a koncovým zařízením. Na začátku zachytávání se obvykle zachytávají pakety, které konfigurují zařízení. Informaci, že jde o klávesnici lze najít v získané odpovědi deskriptoru rozhraní v paketu, který je popsán ve sloupci Info jako GET DESCRIPTOR Response CONFIGURATION. V položce deskriptor rozhraní (Interface Descriptor) je pole s názvem bInterfaceProtocol, které určí, o jaký typ zařízení se jedná.



Obr. 39. bInterfaceProtocol určující typ zařízení,
Zdroj: vlastní zpracování

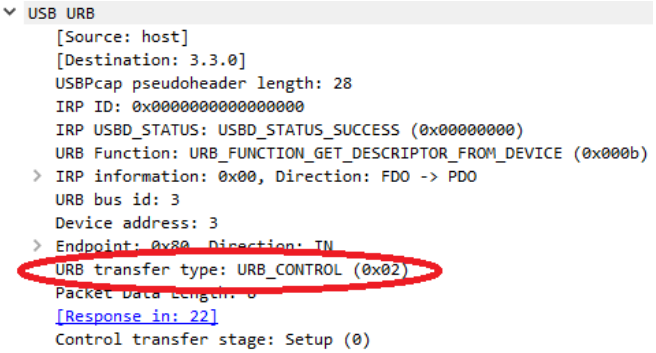
Adresu zařízení pak lze zjistit v položce USB URB ve stejném paketu.



Obr. 40. Adresa zařízení, Zdroj: vlastní zpracování

7.3 Příklad 3. Zjištění typu datového přenosu

Pro příklad byl použit program USBPcap v prostředí Wireshark. Po vybrání paketu, lze typ datového přenosu zjistit v položce USB URB pod názvem URB transfer type. Datový přenos lze zjistit u některých paketů také v sloupci Info.



Obr. 41. Zjištění typu datového přenosu, Zdroj: vlastní zpracování

7.4 Příklad 4. Zachytávání komunikace na klávesnici a myši

Pro příklad byl použit program USBPcap v prostředí Wireshark. Přes analyzátory paketů lze zachytit komunikaci klávesnice nebo myši s hostitelem a to tak, že HID descriptor klávesnice nebo myši posílá zprávy hostiteli. [38]

16	0.000000	2.1.0	host	USB	119 GET DESCRIPTOR Response CONFIGURATION
17	0.000000	host	2.1.0	USB	36 SET CONFIGURATION Request
18	0.000000	2.1.0	host	USB	28 SET CONFIGURATION Response
19	48.787598	2.1.1	host	USB	35 URB_INTERRUPT in
20	48.787718	host	2.1.1	USB	27 URB_INTERRUPT in
21	48.867608	2.1.1	host	USB	35 URB_INTERRUPT in

Obr. 42. První nalezený záznam přenosu přerušení, Zdroj: vlastní zpracování

Komunikace mezi hostitelem a HID zařízeními probíhá s přenosy přerušení, takže je třeba najít tyto pakety. Po zobrazení vybraného paketu lze vidět položku Leftover Capture Data (nebo HID Data), což je právě ta hledaná zpráva obsahující informace stisk klávesy nebo tlačítka myši. Formát zprávy je zapsána hexadecimálně. V případě klávesnice je 1. byte stisknutí modifikátorů (např. Ctrl), 2. byte je prázdný a v 3. bytu jsou vloženy informace o stisknutí běžných kláves. V případě myši 1. byte popisuje stisknutí levého, pravého tlačítka nebo stisknutí kolečka na myši. Zpráva je posílána jak při stisknutí, tak po puštění klávesy (v tu chvíli se objeví zpráva obsahující samé nuly). S nahlédnutím oficiálního dokumentu HID Usage Tables [32] lze pak určit stisknuté klávesy podle tabulky na straně 53.

18	0.000000	2.1.0	host
19	48.787598	2.1.1	host
20	48.787718	host	2.1.1
21	48.867608	2.1.1	host
22	48.867726	host	2.1.1
23	49.467623	2.1.1	host
24	49.467797	host	2.1.1
25	49.507555	2.1.1	host
26	49.507656	host	2.1.1

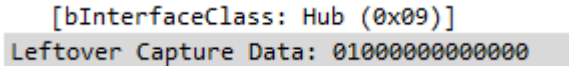
```
> Frame 19: 35 bytes on wire (280 bits), 35 bytes
> USB URB
  Leftover Capture Data: 0000130000000000
```

Obr. 43. Ukázka zprávy, Zdroj: vlastní zpracování

17	11	Keyboard n and N	51	√	√	√	4/101/104
18	12	Keyboard o and O4	25	√	√	√	4/101/104
19	13	Keyboard p and P4	26	√	√	√	4/101/104
20	14	Keyboard q and Q4	17	√	√	√	4/101/104

Obr. 44. Část tabulky z dokumentu HID Usage Tables, Zdroj: vlastní zpracování

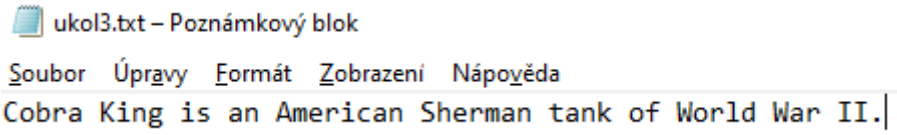
Podle tabulky tedy bylo stisknuta klávesa p.



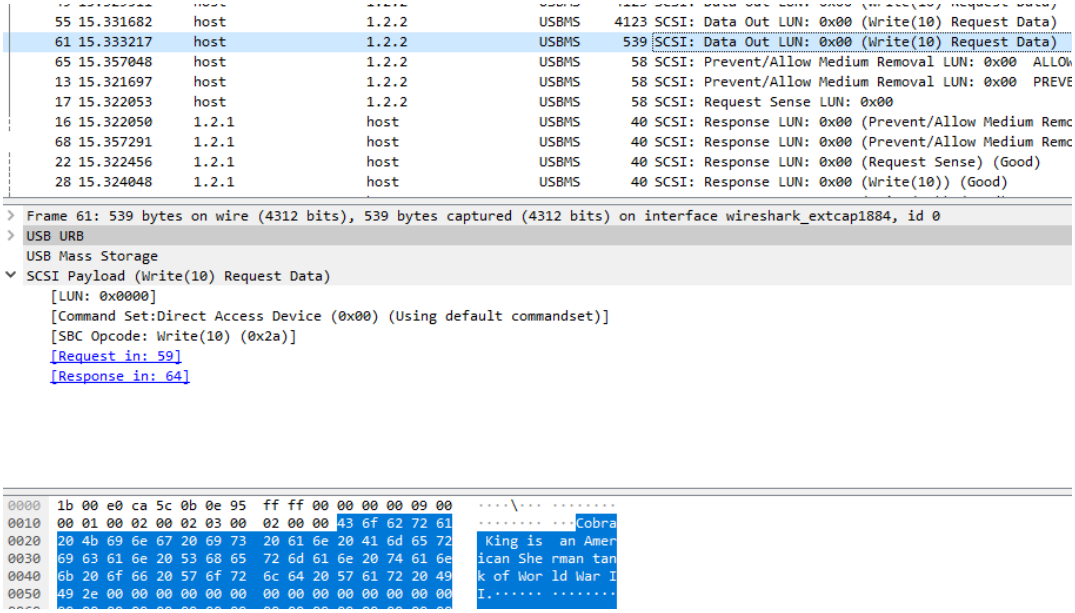
Obr. 45. Stisk levého tlačítka, Zdroj: vlastní zpracování

7.5 Příklad 5. Zachytávání obsahu textu při kopírování

Pro příklad byl použit program USBPcap v prostředí Wireshark. Hromadná uložiště používají hromadné přenosy, ale mohou pro přenos také používat USB Attached SCSI, což je protokol používající standardní sadu SCSI příkazů, který byl představen od standardu USB 3.0. [38] Obsah textu je pak možné nalézt v posledním velkém datovém paketu.



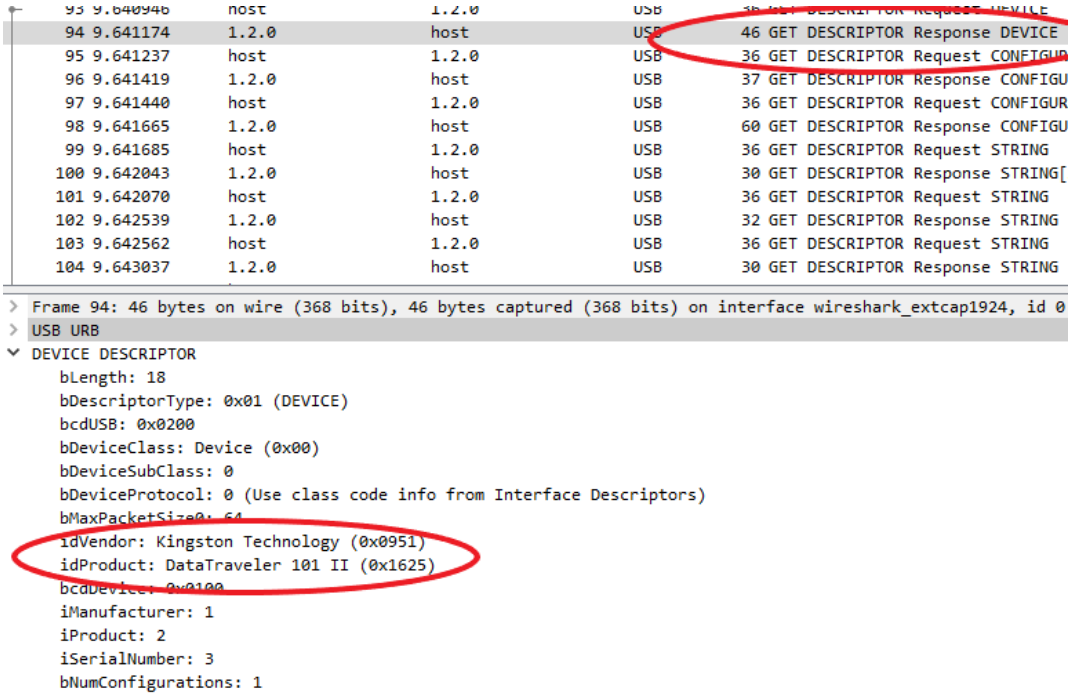
Obr. 46. Textový dokument s větou, Zdroj: vlastní zpracování



Obr. 47. Objevený text při přenosu, Zdroj: vlastní zpracování

7.6 Příklad 6. Identifikace zařízení

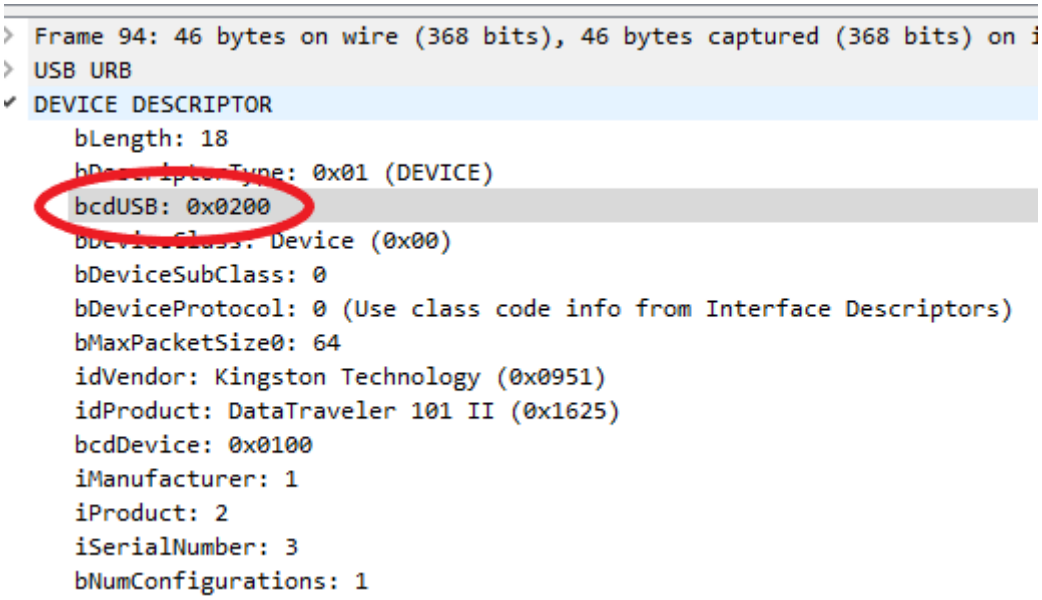
Pro příklad byl použit program USBPcap v prostředí Wireshark. Název výrobce a produktu včetně jejich identifikátorů lze zjistit v deskriptoru zařízení. Hostitel zašle zařízení požadavek GET DESCRIPTOR. Informace o názvu zařízení pak lze nalézt v odpovědi na tento požadavek pod názvy idVendor a idProduct. V závorkách jsou jejich identifikátory.



Obr. 48. Název výrobce i produktu, Zdroj: vlastní zpracování

7.7 Příklad 7. Zjištění verze USB zařízení

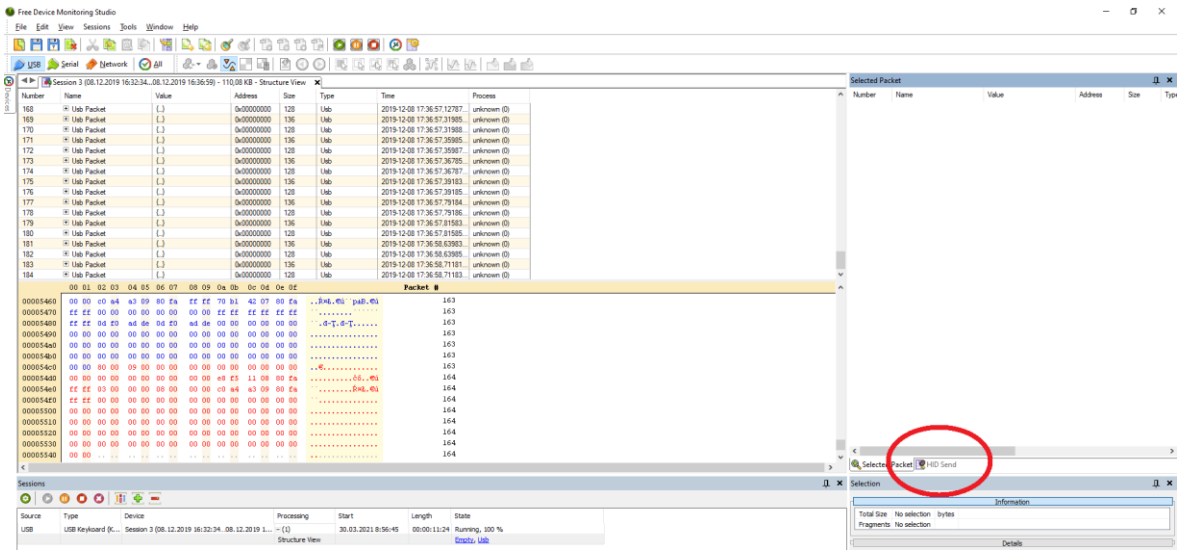
Pro příklad byl použit program USBPcap v prostředí Wireshark. Verze USB lze zjistit v deskriptoru zařízení. Podobně jako v předchozím příkladu, je potřeba nalézt odpověď na požadavek GET DESCRIPTOR. Informace o verzi je pod názvem bcdUSB. 0x0200 znamená, že zařízení je USB 2.0 (0x0300 je USB 3.0 atd.).



Obr. 49. bcdUSB popisuje verzi zařízení, Zdroj: vlastní zpracování

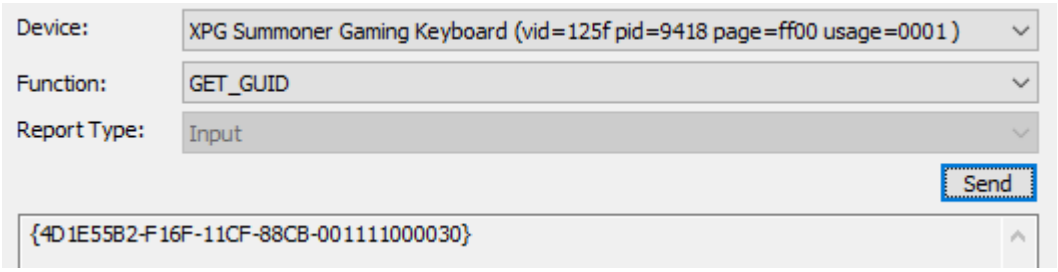
7.8 Příklad 8. Zjistění univerzálního unikátního identifikátoru GUID zařízení

V programu Device monitoring studio je nejprve třeba najít kartu HID Send.



Obr. 50. Karta HID Send, Zdroj: vlastní zpracování

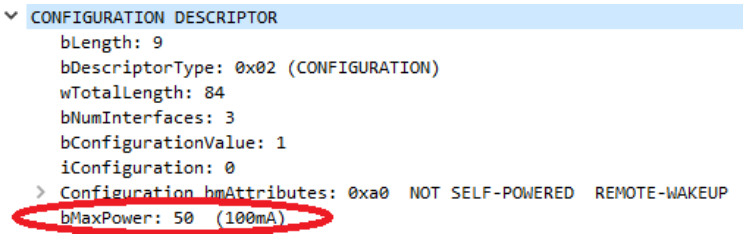
Pak se vybere v kartě zařízení, u kterého je potřeba zjistit GUID a následně se vybere funkce GET_GUID a stiskne tlačítko Send. Výsledné GUID je v tomto případě 4D1E55B2-F16F-11CF-88CB-001111000030.



Obr. 51. Výsledné GUID klávesnice, Zdroj: vlastní zpracování

7.9 Příklad 9. Zjištění maximálního napájení zařízení

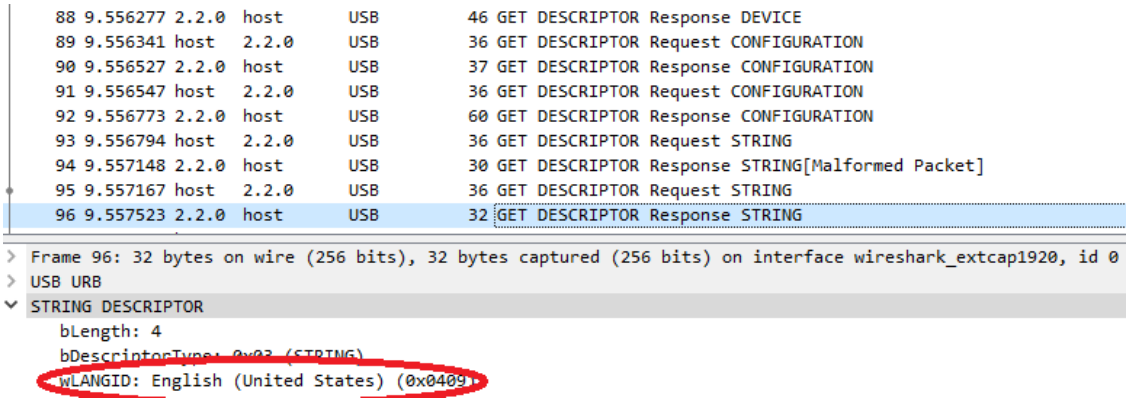
Pro příklad byl použit program USBPcap v prostředí Wireshark. Nejdříve je nutné najít paket obsahující konfigurační deskriptor. To je paket, který má ve sloupci info GET DESCRIPTOR Response CONFIGURATION. V konfiguračním deskriptoru najdeme pole MaxPower, kde udávané číslo je ve 2 mA jednotkách. (v závorkách už převedené).



Obr. 52. Zjištění maximálního napájení zařízení, Zdroj: vlastní zpracování

7.10 Příklad 10. Zjištění jazyka zařízení

Pro příklad byl použit program USBPcap v prostředí Wireshark. Jazyk zařízení lze najít v první odpovědi na požadavek o deskriptor řetězce, tedy paket, který ve sloupci Info má GET DESCRIPTOR Response STRING. Deskriptor řetězce obsahuje pole s názvem wLANGID, který obsahuje jazyk zařízení včetně jeho identifikátoru. Jazyk zařízení lze zjistit i v jiné položce Setup Data v poli s názvem Language Id, ale pouze v následujícím požadavku na deskriptor řetězce. Položka Setup Data objevující se i v předchozích deskriptorech má toto pole nespecifikované, protože deskriptor řetězce se konfiguruje až po nich.



Obr. 53. Zjištění jazyka zařízení, Zdroj: vlastní zpracování

7.11 Příklad 11. Zjištění třídy rozhraní zařízení

Pro příklad byl použit program USBPcap v prostředí Wireshark. Třídu rozhraní zařízení lze najít v získané odpovědi deskriptoru rozhraní v paketu, který je popsán ve sloupci Info jako GET DESCRIPTOR Response CONFIGURATION. V položce deskriptor rozhraní (Interface Descriptor) je pole s názvem bInterfaceClass, které popisuje třídu rozhraní zařízení.

```
INTERFACE DESCRIPTOR (0.0): class Mass Storage
   bLength: 9
   bDescriptorType: 0x04 (INTERFACE)
   bInterfaceNumber: 0
   bAlternateSetting: 0
   bNumEndpoints: 2
   bInterfaceClass: Mass Storage (0x08)
   bInterfaceSubClass: SCSI transparent command set (0x06)
   bInterfaceProtocol: Bulk-Only (BBB) Transport (0x50)
   iInterface: 0
```

Obr. 54. Deskriptor rozhraní, Zdroj: vlastní zpracování

7.12 Příklad 12. Zjištění rozmezí paketů při konfiguraci určitého zařízení a při komunikaci mezi klávesnicí a hostitelem při stisku kláves

Pro příklad byl použit program USBPcap v prostředí Wireshark. Nejprve je třeba zjistit, které pakety patří danému zařízení, což lze udělat přes adresu zařízení. Pro konfiguraci zařízení se používají řídicí přenosy, takže třeba najít všechny pakety s těmito přenosy, které jsou zpravidla u sebe, protože bez konfigurace by nemohla zařízení více komunikovat.

```
13 0.000000 host   3.4.0     USB         36 GET DESCRIPTOR Request DEVICE
14 0.000000 3.4.0  host      USB         46 GET DESCRIPTOR Response DEVICE
15 0.000000 host   3.4.0     USB         36 GET DESCRIPTOR Request CONFIGURATION
16 0.000000 3.4.0  host      USB        119 GET DESCRIPTOR Response CONFIGURATION
17 0.000000 host   3.4.0     USB         36 SET CONFIGURATION Request
18 0.000000 3.4.0  host      USB         28 SET CONFIGURATION Response
```

Obr. 55. Pakety konfigurující zařízení, Zdroj: vlastní zpracování

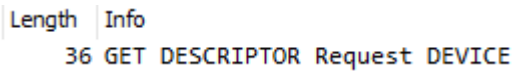
Pro komunikaci stisku kláves se používají přenosy přerušení, takže potřeba najít všechny pakety s těmito přenosy s adresou zařízení, které má klávesnice.

```
275 3.583499 3.4.1  host      USB         35 URB_INTERRUPT in
276 3.583571 host   3.4.1     USB         27 URB_INTERRUPT in
277 3.703509 3.4.1  host      USB         35 URB_INTERRUPT in
278 3.703578 host   3.4.1     USB         27 URB_INTERRUPT in
279 4.231504 3.4.1  host      USB         35 URB_INTERRUPT in
280 4.231575 host   3.4.1     USB         27 URB_INTERRUPT in
281 4.359503 3.4.1  host      USB         35 URB_INTERRUPT in
282 4.359574 host   3.4.1     USB         27 URB_INTERRUPT in
283 5.031484 3.4.1  host      USB         35 URB_INTERRUPT in
284 5.031555 host   3.4.1     USB         27 URB_INTERRUPT in
285 5.111494 3.4.1  host      USB         35 URB_INTERRUPT in
286 5.111565 host   3.4.1     USB         27 URB_INTERRUPT in
287 6.327470 3.4.1  host      USB         35 URB_INTERRUPT in
288 6.327542 host   3.4.1     USB         27 URB_INTERRUPT in
289 6.407474 3.4.1  host      USB         35 URB_INTERRUPT in
290 6.407545 host   3.4.1     USB         27 URB_INTERRUPT in
291 7.263463 3.4.1  host      USB         35 URB_INTERRUPT in
292 7.263533 host   3.4.1     USB         27 URB_INTERRUPT in
293 7.343467 3.4.1  host      USB         35 URB_INTERRUPT in
294 7.343537 host   3.4.1     USB         27 URB_INTERRUPT in
```

Obr. 56. Zachycené pakety při stisku, Zdroj: vlastní zpracování

7.13 Příklad 13. Zjištění délky paketu žádající o deskriptor zařízení

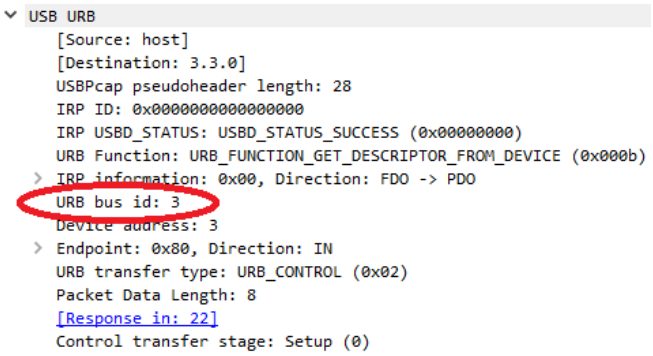
Pro příklad byl použit program USBPcap v prostředí Wireshark. Paket žádající o deskriptor zařízení je takový paket, který má ve sloupci Info GET DESCRIPTOR Request DEVICE. Délku paketu lze zjistit ve sloupci Length. Tento paket má vždy stejnou délku a nezáleží na typu zařízení.



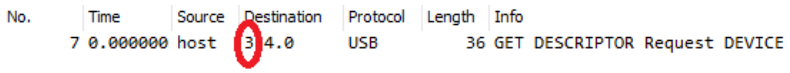
Obr. 57. Délka paketu žádající o deskriptor zařízení, Zdroj: vlastní zpracování

7.14 Příklad 14. Zjištění URB bus id zařízení

Pro příklad byl použit program USBPcap v prostředí Wireshark. Po vybrání paketu, lze URB bus id zjistit buď v položce USB URB nebo ve sloupci Source a Destination, kde první číslo z trojice čísel je URB bus id.



Obr. 58. Zjištění URB bus id v položce USB URB, Zdroj: vlastní zpracování

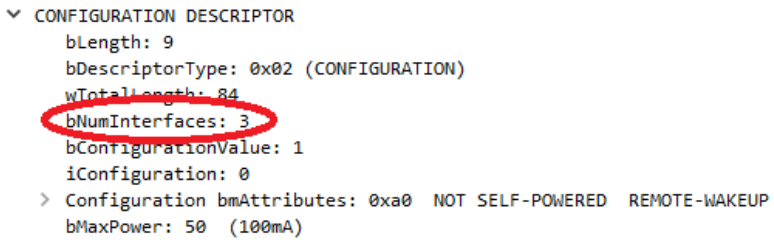


Obr. 59. Zjištění bus id v tabulce paketů, Zdroj: vlastní zpracování

7.15 Příklad 15. Zjištění počtu deskriptorů rozhraní/koncového bodu

Pro příklad byl použit program USBPcap v prostředí Wireshark. Paket obsahující tyto deskriptory má ve sloupci Info GET DESCRIPTOR Response CONFIGURATION.

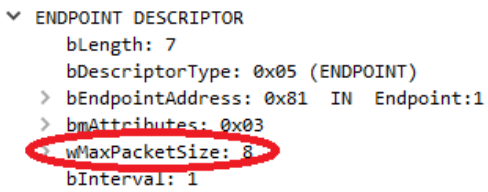
V konfiguračním deskriptoru lze zjistit počet deskriptorů rozhraní podle pole s názvem bNumInterfaces a v deskriptoru rozhraní pak pole s názvem bNumEndpoints.



Obr. 60. Zjištění počtu deskriptorů rozhraní, Zdroj: vlastní zpracování

7.16 Příklad 16. Zjištění maximální velikosti paketu deskriptoru koncového bodu

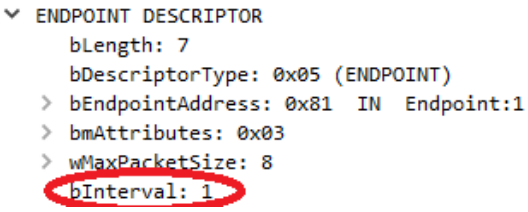
Pro příklad byl použit program USBPcap v prostředí Wireshark. Paket obsahující deskriptor koncového bodu má ve sloupci Info GET DESCRIPTOR Response CONFIGURATION. Po vybrání paketu obsahující deskriptor koncového bodu lze zjistit v položce ENDPOINT DESCRIPTOR pole s názvem wMaxPacketSize informující o maximální velikosti paketu.



Obr. 61. Zjištění maximální velikosti paketu, Zdroj: vlastní zpracování

7.17 Příklad 17. Zjistění intervalu rámců při dotazování koncového bodu zařízení

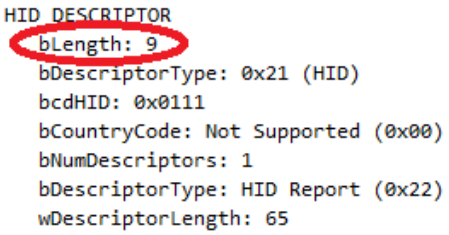
Pro příklad byl použit program USBPcap v prostředí Wireshark. Interval rámců lze najít v paketu obsahující deskriptor koncového bodu, který má ve sloupci Info GET DESCRIPTOR Response CONFIGURATION. Po vybrání paketu lze najít v položce ENDPOINT DESCRIPTOR pole s názvem w bInterval, které má hledanou informaci.



Obr. 62. Zjištění intervalu rámců, Zdroj: vlastní zpracování

7.18 Příklad 18. Zjištění velikosti deskriptoru

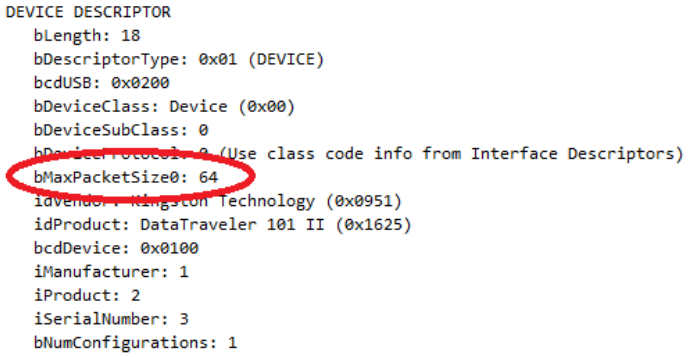
Pro příklad byl použit program USBPcap v prostředí Wireshark. Velikost deskriptoru lze zjistit v paketu obsahující deskriptory. Ve své položce (podle toho, o jaký se deskriptor jedná) je pole s názvem bLength, které určuje velikost deskriptoru.



Obr. 63. Velikost HID deskriptoru, Zdroj: vlastní zpracování

7.19 Příklad 19. Zjištění maximální velikosti paketu pro koncový bod 0

Pro příklad byl použit program USBPcap v prostředí Wireshark. Hledanou informaci je potřeba najít paket obsahující deskriptor zařízení. V tomto deskriptoru pak lze nalézt pole s názvem bMaxPacketSize0 informující o maximální velikosti paketu pro koncový bod 0.



Obr. 64. Zjištění maximální velikosti paketu pro koncový bod 0, Zdroj: vlastní zpracování

ZÁVĚR

V teoretické části byly popsány základní prvky systému USB, jako je např. rozdíl mezi rozbočovačem a koncovým zařízením, jejich fyzická topologie a jejich principy pro průběh komunikace, kterými jsou protokol sběrnice, detekce či zpracování chyb nebo charakterizace zařízení v rámci systémové konfigurace. Byly popsány všechny aktuální konektory a jednotlivé vodiče v průřezu kabelem a rychlosti přenosů jednotlivých verzí Univerzální sériové sběrnice. Bylo vysvětleno, jak v jednotlivých verzí USB probíhá kódování, jednotlivé třídy napájení a kolik je potřeba energie pro USB komunikaci. Dále byla popsána datová komunikace mezi hostitelem a USB zařízením. Byly vysvětleny typy datových přenosů, jejich složení, formát a druhy jednotlivých paketů po vysvětlení jednotlivých paketů. Rovněž byly popsány rozdíly mezi dostupnými verzemi USB.

V praktické části byly vybrány a následně otestovány softwarové sniffery pro zachycení a analýzu komunikace a mezi sebou porovnány. Dále bylo podrobně seznámeno s jedním konkrétním softwarovým snifferem. Po otestování softwarových snifferů byly připraveny vzorové úlohy a jejich řešení pro analýzu USB komunikace různých zařízení. Cílem této práce bylo ze vzorových úloh navrhnout možné zadání a řešení (příloha PI a PII) laboratorní úlohy pro předmět Architektura počítačů.

SEZNAM POUŽITÉ LITERATURY

[1] CUNNINGHAM, Andrew. A brief history of USB, what it replaced, and what has failed to replace it. *Ars Technica* [online]. 2014 [cit. 2020-03-22]. Dostupné z: https://arstechnica.com/gadgets/2014/08/a-brief-history-of-usb-what-it-replaced-and-what-has-failed-to-replace-it/

[2] JANSSEN, Cory. Universal Serial Bus (USB). *Techopedia* [online]. 2012 [cit. 2020-03-20]. Dostupné z: https://www.techopedia.com/definition/2320/universal-serial-bus-usb

[3] JOHNSON, Joel. The unlikely origins of USB, the port that changed everything. *Fast Company | The future of business* [online]. 2019 [cit. 2020-03-22]. Dostupné z: https://www.fastcompany.com/3060705/an-oral-history-of-the-usb

[4] MUELLER, Scott. *Upgrading and Repairing PCs*. 21st Edition. Indianapolis: Pearson Education, © 2013. ISBN 978-0-7897-5000-6.

[5] PERENSON, Melissa J. SuperSpeed USB 3.0: More Details Emerge. *PCWorld - News, tips and reviews from the experts on PCs, Windows, and more* [online]. 2009 [cit. 2020-03-22]. Dostupné z: https://www.pcworld.com/article/156494/superspeed_usb.html

[6] USB IMPLEMENTERS FORUM. *Universal Serial Bus Specification: Revision 2.0* [online]. USB 2.0 Promoter Group, April 27, 2000 [cit. 2020-03-31]. Dostupné z: https://www.usb.org/document-library/usb-20-specification

[7] THOMPSON, Robert Bruce a Barbara FRITCHMAN THOMPSON. *PC Hardware in a Nutshell*. 3rd Edition. O'Reilly Media, 2003. ISBN 059600513X.

[8] AXELSON, Jan. *USB Complete: The Developer's Guide*. Fourth Edition. Madison: Lakeview Research, June 1, 2009. ISBN 978-1-931448-08-6.

[9] USB IMPLEMENTERS FORUM. *Universal Serial Bus 3.2 Specification* [online]. USB 3.0 Promoter Group, September 22, 2017 [cit. 2020-07-05]. Dostupné z: https://www.usb.org/document-library/usb-32-specification-released-september-22-2017-and-ecns

[10] USB IMPLEMENTERS FORUM. *USB Promoter Group Announces USB4 Specification: Specification defines next generation USB protocol architecture and doubling bandwidth to extend USB Type-C™ performance* [online]. USB Promoter Group, March 4, 2019 [cit. 2021-4-24]. Dostupné z:

https://usb.org/sites/default/files/2019-02/USB_PG_USB4_DevUpdate_Announcement_FINAL_20190226.pdf

[11] USB IMPLEMENTERS FORUM. *Universal Serial Bus 4 (USB4™) Specification* [online]. USB 3.0 Promoter Group, October 15, 2020 [cit. 2021-4-24]. Dostupné z: https://www.usb.org/document-library/usb4tm-specification

[12] PEACOCK, Craig. USB in a NutShell - Chapter 4 - Endpoint Types. *Beyondlogic – Electronics Hardware – Embedded Linux – Internet of Things* [online]. Adelaide, c2010 [cit. 2020-07-06]. Dostupné z: https://www.beyondlogic.org/usbnutshell/usb4.shtml

[13] DUYÉ, Bruno. USB_2.0_connectors.svg. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-03-31]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/7/7f/USB_2.0_connectors.svg

[14] MUSIC SORTER. USB_Mini-B.svg. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-03-31]. Dostupné z: https://en.wikipedia.org/wiki/File:USB_Mini-B.svg

[15] FRED THE OYSTER. USB Micro-A connector. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-03-31]. Dostupné z: https://en.wikipedia.org/wiki/File:USB_Micro-A.svg

[16] FRED THE OYSTER. SB Micro-B connector. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-03-31]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/1/15/USB_Micro-B.svg

[17] INGENIEROLOCO. USB 3.0 Type-A plug. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-03-31]. Dostupné z: https://en.wikipedia.org/wiki/File:USB_3.0_Type-A_blue.svg

[18] INGENIEROLOCO. USB B SuperSpeed connector. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-03-31]. Dostupné z: https://en.wikipedia.org/wiki/File:USB_3.0_Type-B_blue.svg

[19] USB IMPLEMENTERS FORUM. *Universal Serial Bus Type-C Cable and Connector Specification* [online]. Release 2.0. USB 3.0 Promoter Group, August

2019 [cit. 2020-3-31]. Dostupné z: https://www.usb.org/document-library/usb-type-cr-cable-and-connector-specification-revision-20-august-2019-and-ecns

[20] INGENIEROLOCO. USB Micro B SuperSpeed connector. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-03-31]. Dostupné z: https://en.wikipedia.org/wiki/File:USB_3.0_Micro-B.svg

[21] CHINDI.AP. USB Type-C plug pinout end-on view. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-03-31]. Dostupné z: https://en.wikipedia.org/wiki/File:USB_Type-C_plug_pinout.svg

[22] *USB 3.1 Legacy Cable and Connector Specification: Revision 1.0 including all errata and ECNs through June 2017* [online]. 2017 [cit. 2020-03-31]. Dostupné z: https://www.usb.org/document-library/usb-31-legacy-cable-and-connector-revision-10

[23] MEDHAT, Maha. Bit stuffing techniques Analysis and a Novel bit stuffing algorithm for Controller Area Network (CAN). *International Journal of Computer System* [online]. March 2015, **2015**(2), 80 [cit. 2020-07-06]. ISSN 2394-1065. Dostupné z: https://www.researchgate.net/publication/329170556_Bit_stuffing_techniques_Analysis_and_a_Novel_bit_stuffing_algorithm_for_Controller_Area_Network_CAN

[24] GARDNER, John V. IBM Customer Engineering Instruction - Reference: 729 II, V, Magnetic Tape Units (30,000 Series). In: *1401 Restoration - CHM* [online]. Smyrna, 1962 [cit. 2020-07-06]. Dostupné z: http://ibm-1401.info/223-6988-729-MagTapeCE-InstRef-62-r.pdf

[25] LEE, Byeong Gi a Seok Chang KIM. *Scrambling Techniques for Digital Transmission: Fundamentals of Scrambling Techniques*. London: Springer, 1994. ISBN 978-1-4471-3231-8. Dostupné z: doi: https://doi.org/10.1007/978-1-4471-3231-8_2

[26] FRANASZEK, Peter A. A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code. *IBM Journal of Research and Development* [online]. September 1983, **1983**(5) [cit. 2021-02-01]. ISSN 0018-8646. Dostupné z: https://hifpga.com/upfiles/15950439066254198.pdf

[27] GOOK, Michael. *PC Hardware Interfaces: A Developer's Reference*. Wayne (Pennsylvania): A-LIST, © 2004. ISBN 193176929X.

[28] PEACOCK, Craig. USB in a NutShell - Chapter 3 - USB Protocols. *Beyondlogic – Electronics Hardware – Embedded Linux – Internet of Things* [online]. Adelaide, c2010 [cit. 2020-07-06]. Dostupné z: https://www.beyondlogic.org/usbnutshell/usb3.shtml

[29] USB Data Packet Structure: Technical NoteTN_116. *FTDI Chip* [online]. Glasgow, 2009 [cit. 2020-07-06]. Dostupné z: https://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_116_USB%20Data%20Structure.pdf

[30] MOŃ, Tomasz. USBPcap Capture format specification. *Open Source USB Packet capture for Windows* [online]. [cit. 2021-4-29]. Dostupné z: https://desowin.org/usbpcap/captureformat.html

[31] MOŃ, Tomasz. USBPcap - USB Packet capture for Windows. *Open Source USB Packet capture for Windows* [online]. [cit. 2021-4-29]. Dostupné z: https://desowin.org/usbpcap/index.html

[32] Universal Serial Bus (USB): HID Usage Tables. In: *Implementers' Forum* [online]. 2004 [cit. 2020-07-09]. Dostupné z: https://www.usb.org/sites/default/files/documents/hut1_12v2.pdf

[33] HHD SOFTWARE. *Free Serial Analyzer* [online]. [cit. 2021-4-29]. Dostupné z: https://freeserialanalyzer.com/features

[34] USBLYZER TEAM. *USB Analysis Features of USBlyzer* [online]. [cit. 2021-4-29]. Dostupné z: http://www.usblyzer.com/usb-analysis-features.htm

[35] MOŃ, Tomasz. USB Packet capture for Windows Tour. *Open Source USB Packet capture for Windows* [online]. [cit. 2021-4-29]. Dostupné z: https://desowin.org/usbpcap/tour.html

[36] WIRESHARK FOUNDATION. *Wireshark User's Guide* [online]. [cit. 2021-4-29]. Dostupné z: https://www.wireshark.org/docs/wsug_html_chunked/

[37] STONER, Ronald. Hackmethod July 2017 — Challenges Write Up. *Medium* [online]. San Francisco: A Medium Corporation, 2017, 1 Aug 2017 [cit. 2021-5-9].

Dostupné z: https://medium.com/@forwardsecrecy/hackmethod-july-2017-challenges-write-up-1303f414c8d6

[38] USB IMPLEMENTERS FORUM. *Universal Serial Bus (USB): Device Class Definition for Human Interface Devices (HID)* [online]. c1996-2001 [cit. 2021-5-10]. Dostupné z: https://www.usb.org/sites/default/files/hid1_11.pdf

[39] USB IMPLEMENTERS FORUM. *Universal Serial Bus Mass Storage Class: USB Attached SCSI Protocol (UASP)* [online]. Revision 1.0. June 24, 2009 [cit. 2021-02-01]. Dostupné z: https://www.usb.org/document-library/usb-attached-scsi-protocol-uasp-v10-and-adopters-agreement

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ACK	Acknowledge packet,
ADDR	Address Field
CRC	Cyclic Redundancy Checks
DMS	Device Monitoring Studio
ENDP	Endpoint Field
EOP	End-of-Packet
ERDY	Endpoint Ready
ERR	Split Transaction Error Handshake
FIFO	First in, first out
HID	Human interface device
LFSR	Linear-feedback shift register
LSB	Least significant bit
NAK	No Acknowledge packet,
NRDY	Not Ready
NRZI	Non-return-to-zero inverted
NYET	No response yet from receiver
PID	Packet ID
PRE	Host-issued preamble
SOF	Start-of-Frame
SYNC	Synchronization
USB	Universal Serial Bus
USB URB	USB Request Block
IRP	I/O Request packet

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: VZOROVÁ LABORATORNÍ ÚLOHA - ZADÁNÍ

Úkol 1. Převeďte prvních 5 písmen svého jména podle kódovací tabulky ASCII na binární kód, aplikujte kódování NRZI-S (včetně bit stuffingu, pokud nutno), které používá USB 2.0 pro své kódování, a graficky znázorněte.

Úkol 2. Spusťte program Wireshark a zvolte zachytávání kořenového rozbočovače obsahující HID zařízení (klávesnice, myš). V průběhu zachytávání napište prvních 5 písmen svého jména, použijte minuskule (malá písmena). Zjistěte a doložte pod každý úkol výstřižek obrazovky:

1. Adresu zařízení (Device address) klávesnice

2. Rozmezí paketů, kde došlo ke komunikaci mezi klávesnicí a hostitelem při stisku kláves

3. Délku libovolného paketu v Bytech, který žádá o deskriptor zařízení (device descriptor)

4. Znaky zastupující prvních 5 písmen vašeho jména odchycených z paketů zapsané v hexadecimální soustavě

5. Uveďte typ datového přenosu jednoho z vašich paketů zachycených při stisku kláves

6. Název výrobce zařízení

7. VendorID a ProductID zařízení

8. Verzi USB připojeného zařízení

9. Třídu rozhraní zařízení

10. URB bus id zařízení

11. Počet deskriptorů rozhraní

12. Maximální velikost paketu deskriptoru (nebo deskriptorů, pokud je jich víc) koncového bodu

Úkol 3. Vyberte si vlastní libovolné zařízení, připojte ho do systému a spusťte odchytávání. Zjistěte:

1. Název produktu

2. VendorID a ProductID zařízení

3. Verzi USB připojeného zařízení

4. Maximální napájení zařízení

5. Jazyk zařízení

6. Třídu rozhraní zařízení

7. Uveďte všechny typy datového přenosu, které vaše zařízení používá

8. Maximální velikost paketu pro koncový bod 0

9. Počet koncových bodů používající rozhraní zařízení

10. Jaký je interval rámců při dotazování koncového bodu zařízení

11. Velikost konfiguračního deskriptoru

12. Napište rozmezí paketů, které sniffer zachytil při rozpoznání a konfiguraci vašeho zařízení

PŘÍLOHA P II: VZOROVÁ LABORATORNÍ ÚLOHA – ŘEŠENÍ

Úkol 1. Převeďte prvních 5 písmen svého jména podle kódovací tabulky ASCII na binární kód, aplikujte kódování NRZI-S (včetně bit stuffingu, pokud nutno), které používá USB 2.0 pro své kódování, a graficky znázorněte.

Marti

65 97 114 116 105

01001101 01100001 01110010 01110100 01101001

00100011 00010100 11110110 11110010 11100100

Úkol 2. Spusťte program Wireshark a zvolte zachytávání kořenového rozbočovače obsahující HID zařízení (klávesnice, myš). V průběhu zachytávání napište prvních 5 písmen svého jména, použijte minuskule (malá písmena). Zjistěte a doložte pod každý úkol výstřižek obrazovky:

1. Adresu zařízení (Device address) klávesnice

```
Device address: 4
```

2. Rozmezí paketů, kde došlo ke komunikaci mezi klávesnicí a hostitelem při stisku kláves

```
275 3.583499 3.4.1  host      USB         35 URB_INTERRUPT in
276 3.583571 host   3.4.1     USB         27 URB_INTERRUPT in
277 3.703509 3.4.1  host      USB         35 URB_INTERRUPT in
278 3.703578 host   3.4.1     USB         27 URB_INTERRUPT in
279 4.231504 3.4.1  host      USB         35 URB_INTERRUPT in
280 4.231575 host   3.4.1     USB         27 URB_INTERRUPT in
281 4.359503 3.4.1  host      USB         35 URB_INTERRUPT in
282 4.359574 host   3.4.1     USB         27 URB_INTERRUPT in
283 5.031484 3.4.1  host      USB         35 URB_INTERRUPT in
284 5.031555 host   3.4.1     USB         27 URB_INTERRUPT in
285 5.111494 3.4.1  host      USB         35 URB_INTERRUPT in
286 5.111565 host   3.4.1     USB         27 URB_INTERRUPT in
287 6.327470 3.4.1  host      USB         35 URB_INTERRUPT in
288 6.327542 host   3.4.1     USB         27 URB_INTERRUPT in
289 6.407474 3.4.1  host      USB         35 URB_INTERRUPT in
290 6.407545 host   3.4.1     USB         27 URB_INTERRUPT in
291 7.263463 3.4.1  host      USB         35 URB_INTERRUPT in
292 7.263533 host   3.4.1     USB         27 URB_INTERRUPT in
293 7.343467 3.4.1  host      USB         35 URB_INTERRUPT in
294 7.343537 host   3.4.1     USB         27 URB_INTERRUPT in
```

3. Délku libovolného paketu v Bytech, který žádá o deskriptor zařízení (device descriptor)

```
Length  Info
    36  GET DESCRIPTOR Request DEVICE
```

4. Znaky zastupující prvních 5 písmen vašeho jména odchycených z paketů zapsané v hexadecimální soustavě

```
HID Data: 0000100000000000
HID Data: 0000040000000000
HID Data: 0000150000000000
HID Data: 0000170000000000
HID Data: 00000c0000000000
```

5. Uveďte typ datového přenosu jednoho z vašich paketů zachycených při stisku kláves

```
URB transfer type: URB_INTERRUPT (0x01)
```

6. Název výrobce zařízení

```
idVendor: SiGma Micro
```

7. VendorID a ProductID zařízení

```
idVendor: SiGma Micro (0x1c4f)
idProduct: Keyboard TRACER Gamma Ivory (0x0002)
```

8. Verzi USB připojeného zařízení

```
bcdUSB: 0x0110
```

9. Třídu rozhraní zařízení

```
bInterfaceClass: HID (0x03)
```

10. URB bus id zařízení

```
URB bus id: 3
```

11. Počet deskriptorů rozhraní

```
bNumInterfaces: 2
```

12. Maximální velikost paketu deskriptoru (nebo deskriptorů, pokud je jich víc) koncového bodu

```
> wMaxPacketSize: 8
> wMaxPacketSize: 3
```

Úkol 3. Vyberte si vlastní libovolné zařízení, připojte ho do systému a spusťte odchytávání. Zjistěte:

1. Název produktu

```
idProduct: DataTraveler 101 II
```

2. VendorID a ProductID zařízení

```
idVendor: Kingston Technology (0x0951)
idProduct: DataTraveler 101 II (0x1625)
```

3. Verzi USB připojeného zařízení

```
bcdUSB: 0x0200
```

4. Maximální napájení zařízení

```
bMaxPower: 50  (100mA)
```

5. Jazyk zařízení

```
wLANGID: English (United States) (0x0409)
```

6. Třídu rozhraní zařízení

```
bInterfaceClass: Mass Storage (0x08)
```

7. Uveďte všechny typy datového přenosu, které vaše zařízení používá

```
URB transfer type: URB_CONTROL (0x02)
URB transfer type: URB_BULK (0x03)
```

8. Maximální velikost paketu pro koncový bod 0

```
bMaxPacketSize0: 64
```

9. Počet koncových bodů používající rozhraní zařízení

```
bNumEndpoints: 2
```

10. Jaký je interval rámců při dotazování koncového bodu zařízení

```
bInterval: 255
```

11. Velikost konfiguračního deskriptoru

```
bLength: 9
```

12. Napište rozmezí paketů, které sniffer zachytil při rozpoznání a konfiguraci vašeho zařízení

```
 87 5.494229 host   2.2.0     USB        36 GET DESCRIPTOR Request DEVICE
 88 5.494508 2.2.0  host      USB        46 GET DESCRIPTOR Response DEVICE
 89 5.494553 host   2.2.0     USB        36 GET DESCRIPTOR Request CONFIGURATION
 90 5.494755 2.2.0  host      USB        37 GET DESCRIPTOR Response CONFIGURATION
 91 5.494770 host   2.2.0     USB        36 GET DESCRIPTOR Request CONFIGURATION
 92 5.495000 2.2.0  host      USB        60 GET DESCRIPTOR Response CONFIGURATION
 93 5.495043 host   2.2.0     USB        36 GET DESCRIPTOR Request STRING
 94 5.495500 2.2.0  host      USB        30 GET DESCRIPTOR Response STRING[Malformed Packet]
 95 5.495537 host   2.2.0     USB        36 GET DESCRIPTOR Request STRING
 96 5.496001 2.2.0  host      USB        32 GET DESCRIPTOR Response STRING
 97 5.496023 host   2.2.0     USB        36 GET DESCRIPTOR Request STRING
 98 5.496500 2.2.0  host      USB        30 GET DESCRIPTOR Response STRING
 99 5.496546 host   2.2.0     USB        36 GET DESCRIPTOR Request STRING
100 5.497003 2.2.0  host      USB        78 GET DESCRIPTOR Response STRING
101 5.497849 host   2.2.0     USB        36 SET CONFIGURATION Request
102 5.498072 2.2.0  host      USB        28 SET CONFIGURATION Response
103 5.498115 host   2.2.0     USB        36 SET INTERFACE Request
104 5.508285 2.2.0  host      USB        28 SET INTERFACE Response
105 5.508339 host   2.2.0     USBMS      36 GET MAX LUN Request
106 5.508537 2.2.0  host      USBMS      29 GET MAX LUN Response
```