VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

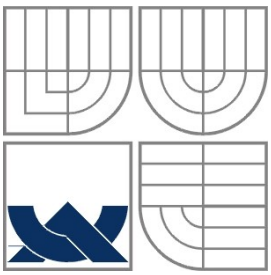
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

USB HOST S MIKROKONTROLÉRY PIC

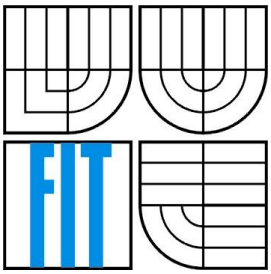
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. PAVEL KUČERA
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

USB HOST S MIKROKONTROLÉRY PIC

USB HOST WITH PIC MICROCONTROLLERS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL KUČERA

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Dr. Ing. PETR HANÁČEK

BRNO 2010

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2009/2010

Zadání diplomové práce

Řešitel: **Kučera Pavel, Bc.**
Obor: Počítačové systémy a sítě
Téma: **USB host s mikrokontroléry PIC**
USB Host with PIC Microcontrollers
Kategorie: Počítačové sítě

Pokyny:

1. Prostudujte komunikační protokol USB.
2. Seznamte se s vývojovým kitem Microchip pro USB host.
3. Navrhněte firmware aplikace pro připojení sériového zařízení (modem, telefon GSM) pomocí USB k mikrokontroléru PIC.
4. Navržené řešení implementujte.
5. Zhodnoťte dosažené výsledky.

Literatura:

- Don Anderson: USB System Architecture (USB 2.0), MINDSHARE, INC.
- Dle pokynů vedoucího.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 - 3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hanáček Petr, doc. Dr. Ing.**, UITS FIT VUT
Datum zadání: 21. září 2009
Datum odevzdání: 26. května 2010

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Předmětem diplomové práce je implementace USB hostitele v mikrokontroléru PIC. Práce obsahuje popis vývoje způsobu připojování zařízení k počítači se zaměřením na detailní popis sběrnice USB. Jsou popsány způsoby řešení USB hostitelů pro vestavěné systémy. Na základě analýzy současného stavu je navržen systém USB hostitele s rozdělením aplikace do vrstev, který je následně implementován v mikrokontroléru. Výsledný firmware USB hostitele je testován na vývojovém kitu s připojením reálného zařízení.

Abstract

The aim of master thesis is implementation of USB host in a microcontroller PIC. The thesis contains description of ways of peripherals to computer connections with detailed description of USB bus. There are also discussed several approaches in implementations USB host for embedded systems. A concept of layered USB host system is designed on the basis of analysis of the state of the art. Finally, the designed system is implemented in microcontroller. Resulting firmware of development kit is tested on real USB device.

Klíčová slova

USB host, sběrnice, koncový bod, přenos, mikrokontrolér, mezivrstva, API, HAL, Microchip, PIC, MPLAB, komunikační zařízení, modem, mobilní telefon.

Keywords

USB host, bus, endpoint, transfer, microcontroller, layer, API, HAL, Microchip, PIC, MPLAB, communication device, modem, cell phone.

Citace

Pavel Kučera: USB host s mikrokontroléry PIC, diplomová práce, Brno, FIT VUT v Brně, 2010

USB host s mikrokontroléry PIC

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Doc. Dr. Ing. Petra Hanáčka.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Pavel Kučera
26.5.2010

Poděkování

Děkuji tímto vedoucímu diplomové práce, panu Doc. Dr. Ing. Petru Hanáčkovi, za cenné připomínky, zejména ke koncepci textu práce. Firmě ADCIS, s.r.o. děkuji za zapůjčení vývojového kitu Atmel EVK1100.
Děkuji rodině za podporu a partnerce za trpělivost během mého celého studia.

Obsah

1 Úvod

Diplomová práce má seznámit čtenáře s problematikou vytváření aplikace pro mikrokontroléry využívající rozhraní USB, konkrétně USB hostitele pro použití v mikrokontroléru rodiny PIC24F. Zabývá se problematikou vzniku USB rozhraní a USB protokolu v kontextu dalších rozhraní periferních zařízení.

V rámci seznámení se s USB protokolem jsou vysvětleny základní pojmy, principy komunkace na sběrnici, mimo jiné jsou vysvětleny pojmy USB host a USB zařízení. Po objasnění uvedených pojmů následuje analýza současného stavu, popisuje výhody a nevýhody jednotlivých existujících řešení USB hostitelů se zaměřením na USB rozhraní ve vestavěných zařízeních. Na základě analýzy současného stavu a existujících možných přístupů k řešení problematiky USB hostitele je navržen systém pro firmware mikroprocesoru PIC, ten je následně v souladu s návrhem implementován, přičemž jsou popsány konstrukce významných funkčních částí.

Následuje popis různých přístupů k testování systému na různých úrovních během jeho implementace a testování výsledné aplikace. V závěru jsou shrnuty a zhodnoceny dosažené výsledky práce, včetně určení možného využití aplikace a možná rozšíření aplikace v budoucnosti.

2 Přehled současného stavu

Kapitola vysvětluje základní pojmy jako rozhraní, periferní zařízení, principy komunikace, princip ovládání periferního zařízení procesorem počítače a vzájemný vztah komunikujících prvků. Popisuje současný stav problematiky připojování periferních zařízení v kontextu s historií, zmiňuje rozšířené standardy, výhody a nevýhody jednotlivých řešení. Z rozšířených standardů zmiňuje zejména sériový a paralelní port (RS-232 a LPT), z modernějších rozhraní potom IrDA, USB, firewire, bluetooth a eSATA. Následuje detailní popis USB, který se zabývá koncepčním řešením protokolu a rozhraní, osvětluje základy principu komunikace na sběrnici USB včetně popisu jednotlivých verzí s důrazem na verzi standardu USB 2.0. Kapitolu zakončuje detailní pohled na USB hostitele s popisem jednotlivých standardů a použitých řešení u konkrétních produktů.

2.1 Základní pojmy

Dle [1] je *rozhraní* definováno jako obecné ohraničení sdílené dvěma systémy, zařízeními nebo programy, stejně jako prvky tohoto propojovacího rámce a přídavné obvody sloužící pro propojování zařízení, které slouží k jejich vzájemnému propojení a komunikaci. Z výše uvedeného je patrné, že pod pojmem rozhraní si můžeme představit jak záležitost komunikace na softwarové úrovni (instrukce počítačového programu), tak na hardwarové úrovni (hodnoty napětí reprezentující logické hodnoty přenášených dat, topologii), proto v dalším textu bude v místech, kde to bude pro pochopení textu vhodné, rozhraní explicitně rozlišeno na softwarové rozhraní a fyzické rozhraní. Pod pojmem *periferní zařízení* se obecně skrývají všechny komponenty mimo výpočetní jádro (procesor, hlavní paměť a prostředky sloužící k jejich propojení). Tato zařízení je možno rozdělit do tří základních skupin – datová úložiště, vstup-výstupní zařízení a komunikační zařízení. Datová úložiště jsou pevné disky, optické mechaniky, pásková zařízení, paměťové karty, flash disky a podobně. Vstup-výstupní zařízení jsou zařízení, která převádí informaci z vnitřní datové reprezentace počítače do podoby srozumitelné člověku nebo jiným technickým prostředkům a naopak. Do této skupiny zařízení patří polohovací zařízení (klávesnice, myš), tiskárny, skenery, mikrofony a mnoho dalších. Poslední skupinou jsou komunikační zařízení, ty slouží k přenosu informace mezi počítači nebo jejich částmi. Patří sem modemy, síťové karty, převodníky a další. Nebude-li uvedeno jinak, bude se v dalším textu za periferní zařízení považovat pouze zařízení vně počítače.

Softwarová rozhraní jsou zejména u PC kompatibilních zařízení vytvářena na úrovni registrů. Registry si je možné v analogii běžného života představit jako poštovní schránky, do kterých jsou umisťovány zprávy. Registry jsou v systému jednoznačně určeny svojí adresou. Fyzické adresy registrů se dříve mapovaly výhradně do vstupně výstupního prostoru (I/O space), jehož velikost je u procesorů řady x86 64 kB. Při mapování fyzických adres do vstupně výstupního prostoru docházelo k zarovnávání adres pro zachování jednoduchého a rychlého přístupu ze strany procesoru Tento prostor byl adresově oddělen od adresního prostoru hlavní paměti a přistupovalo se do něj pomocí speciálních instrukcí. V průběhu vývoje přestal tento prostor stačit, protože se zvyšoval počet a složitost zařízení, které procesor musel obsluhovat. Z tohoto důvodu se registry periferních zařízení začaly mapovat do adresního prostoru hlavní paměti, který nebyl využit fyzickou pamětí RAM. V dnešní době je nejrozšířenější způsob adresování všech zařízení v rámci adresního prostoru hlavní paměti, tedy včetně vstupně výstupního prostoru. Ať už je použit jakýkoliv princip adresování, vždy

je třeba bezpodmínečně zajistit konzistenci, jedna adresa v paměti musí vždy ukazovat pouze na jediné fyzické zařízení, na jediný registr.

Fyzická rozhraní je možno klasifikovat a porovnávat podle řady různých kritérií, mezi které patří zejména způsob přenosu dat po přenosovém médiu, druh přenosového média, přenosová rychlost, způsob řízení komunikace a vzdálenost komunikujících prvků. Co se týče přenosu dat po přenosovém médiu, je možné rozlišovat přenos sériový nebo paralelní a synchronní nebo asynchronní. U paralelního přenosu jsou přenášeny všechny bity v rámci skupiny (bajt, slovo, dvojslovo) v jeden okamžik, každý bit je přenášen po separátním fyzickém kanálu, zpravidla po vodiči, u sériového přenosu jsou přenášeny bity po sobě v časovém sledu po společném fyzickém kanálu. Přenos je synchronní, pokud kromě kanálu pro data je v rozhraní vyčleněn kanál pro signalizaci synchronizace, který určuje, kdy jsou data na datovém kanálu platná a lze tedy přečíst jejich hodnotu. Asynchronní přenos nemá vyčleněný kanál pro synchronizaci, v takovém případě se synchronizace získává z přenášených dat. Přenosovým médiem může být vodič elektrického proudu, optické vlákno nebo vzduch. Při použití vzduchu jako přenosového média hovoříme o *bezdrátovém rozhraní*. Přenosová rychlost udává, jak velké množství dat je možné přes rozhraní přenést za jednotku času. Způsob řízení komunikace je buď hierarchický (tzv. master-slave), nebo rovnocenný (tzv. peer-to-peer). U hierarchického způsobu řízení komunikace je vždy jeden prvek v rámci rozhraní řídicí, ostatní prvky se podřizují tomuto prvku, zatímco u rovnoceného způsobu řízení komunikace jsou si prvky v rozhraní rovny a komunikují na přenosovém médiu podle předem jasně daných pravidel nebo o přenosové médium soutěží.

2.2 Přehled existujících rozhraní

Rozhraní jsou dnes téměř výhradně široce standardizována, protože se to v průběhu vývoje výpočetních prostředků ukázalo jako výhodné. Dříve (přibližně do 70.-80. let 20. století) byla standardizace rozhraní často pouze na úrovni jednotlivých výrobců. Každý výrobce používal různé druhy konektorů, různé protokoly a jedinou možností jak připojit periferní zařízení jednoho výrobce k počítači druhého výrobce byla instalace řadičové karty, která fungovala jako převodník mezi rozhraními. Problém tohoto řešení byl nejen v pracnosti, kdy bylo potřeba počítač vypnout, oddělat kryt, nainstalovat kartu, ručně ji nakonfigurovat, vrátit kryt, zapnout počítač a nainstalovat softwarový ovladač, ale i v ceně takového řešení a možnostech rozšiřitelnosti. Cena byla vysoká z důvodu potřeby dalšího, dalo by se říci nadbytečného (byť nezbytného), hardwaru. Dokonce nebyly výjimkou situace, kdy rozšiřující karty nebyly pro danou kombinaci vnitřního a vnějšího rozhraní dostupné a bylo třeba je kompletně vyvinout. V té době navíc byly standardy nedostupným, bedlivě střeženým zbožím výrobců, proto mnohdy nezbylo nic jiného, než využít reverzního inženýrství. Rozšiřitelnost byla výrazně omezená, protože řadičových karet bylo možné do počítače fyzicky instalovat pouze velmi omezené množství, řada z nich byla navíc nezbytně nutná pro obsluhu počítače (výstup na monitor, vstup klávesnice), o rozšíření se potom nedalo moc mluvit.

V souvislosti se vznikem a s výrazným rozšířením osobních počítačů v průběhu 80. let 20. století došlo k výraznému zlepšení. Výrobců bylo stále mnoho, kompatibilita však již hrála poměrně důležitou roli, zejména pak u menších značek. V té době se téměř v každém osobním počítači vyskytuje sériové a paralelní rozhraní známé pod označením *COM* a *LPT*. Zmíněná rozhraní nebyla v té době nová, díky masovému rozšíření se však staly standardem pro připojování periferních

zařízení k počítači. Stále jsou však problémy se snadnou konfigurovatelností a pohodlím obsluhy při manipulaci s těmito zařízeními, zařízení nelze připojit za běhu počítače, konfiguraci je potřeba provádět často ručně, konektorů je omezený počet, přenosová rychlost v řádu desítek kB/s nestačí.

Výše zmíněné problémy vedly v polovině 90. let 20. století ke vzniku rozhraní *USB* a *FireWire* (IEEE 1394). Společné vlastnosti těchto rozhraní zahrnují možnost připojení zařízení za běhu počítače, automatickou konfiguraci, připojení více zařízení k jednomu fyzickému konektoru v počítači, možnost napájet zařízení ze sběrnice a vyšší přenosovou rychlost než doposud používaná rozhraní. Přenosová rychlost je řádově v jednotkách MB/s. Přestože rozhraní FireWire má proti USB některé jasné výhody jako vyšší přenosovou rychlost, větší možnosti napájení, nižší zatížení procesoru počítače, není zdaleka tak rozšířené, jako USB. Důvodem je nutná větší složitost samotných zařízení pro rozhraní FireWire, které zvyšuje jejich cenu, proto se FireWire uplatňuje zejména v aplikacích, kde je kladen velký důraz na přenosovou rychlost jako datová úložiště, videokamery a podobně.

Rozhraní	**Využití**	**Rychlost**	**Dosah**
COM (RS-232)	komunikační zařízení, myši, tiskárny, průmysl	až 115,2 kb/s (11,5 kB/s)	15 m
LPT (IEEE 1284)	tiskárny, skenery, plottery	12 kb/s (1,5 kB/s)	10 m
IrDA	mobilní telefony, PDA, notebooky	až 115,2/4000 kb/s (11,5/400 kB/s) pro SIR/FIR přenos	1 m
FireWire (IEEE 1394)	datová úložiště, kamery, spotřební elektronika	400 Mb/s až 3,2 Gb/s (50 MB/s až 800 MB/s)	4,5 m pro spoj, 72 m pro celou sběrnici
USB 1.0, USB 1.1	různá periferní zařízení	1,5 až 12 Mb/s (200 kB/s až 1,5 MB/s)	3 m
USB 2.0	různá periferní zařízení	až 480 Mb/s (60 MB/s)	5 m pro spoj, 30 m pro celou sběrnici
USB 3.0	různá periferní zařízení, spíše datová úložiště, kamery	až 5 Gb/s (600 MB/s)	5 m pro spoj, 30 m pro celou sběrnici
Bluetooth (IEEE 802.15.1)	mobilní zařízení, PDA, notebooky, handsfree	1 až 24 Mb/s (100 kB/s až 3 MB/s)	až 100 m
eSATA	datová úložiště	3 Gb/s (300 MB/s)	2 m

Tabulka 2.1: Přehledová tabulka rozhraní periferních zařízení

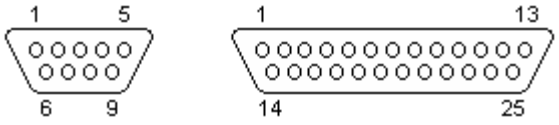
S pokrokem technologií rostly nároky na přenosovou rychlost, proto po roce 2000 vznikaly a stále vznikají nové standardy, avšak s důrazem na zpětnou kompatibilitu. Vzniká nový standard jak

USB, tak FireWire, přenosové rychlosti se zvyšují na desítky MB/s při zachování stejných konektorů. V době vzniku této práce poslední verze zmíněných rozhraní dosahují přenosových rychlostí v řádu stovek MB/s. Specifickým rozhraním je eSATA, které bylo standardizováno v roce 2004. Jde o externí verzi rozhraní SATA, rozhraní je užíváno výhradně pro připojování datových úložišť a optických mechanik. Přenosová rychlost rozhraní eSATA je řádově ve stovkách MB/s.

Z bezdrátových rozhraní bych rád zmínil IrDA a Bluetooth. IrDA využívá k přenosu infračerveného světla, Bluetooth využívá rádiových vln. Vznik obou rozhraní spadá přibližně do poloviny 90. let 20. století. Přenosové rychlosti jsou v řádu jednotek až stovek kB/s. Výhodou IrDA ve srovnání s Bluetooth je nižší cena zařízení, nevýhodou potom potřeba přímé viditelnosti a krátký dosah (cca 1m).

2.2.1 COM

Rozhraní je též známo jako sériové rozhraní. Největšího rozšíření rozhraní dosáhlo s příchodem počítačů řady IBM PC/AT. V 80. a 90. letech 20. století osobní počítače obsahovaly zpravidla jeden nebo dva porty tohoto rozhraní. V dnešní době se od tohoto rozhraní u osobních počítačů upouští ve prospěch novějších (USB, FireWire), v průmyslových aplikacích je rozhraní stále velice rozšířeno pro svoji jednoduchost a nenáročnost implementace na softwarové i hardwarové úrovni. Na fyzické úrovni rozhraní implementuje standard RS-232 z roku 1969 (existují i rozšíření z pozdějších let), jde o sériové asynchronní rozhraní, které bylo zpravidla užíváno pro připojování polohovacích zařízení jako počítačové myši nebo joysticky nebo komunikačních zařízení jako modemy, terminály případně pro připojování tiskáren a podobně. Nejčastěji je užíváno 9 pinového konektoru DE-9 (D-sub 9) nebo 25 pinového DB-25 typu samec. Z hlediska systému nemusí jít jen o skutečné fyzické rozhraní, ale jako COM mohou být mapována různá virtuální zařízení nebo zařízení jiných fyzických rozhraní, například převodníky do rozhraní USB nebo jiných. Na softwarové úrovni je COM implementováno jako sedm osmibitových registrů. Registry slouží pro přenos dat a pro konfiguraci přenosových rychlostí a dalších parametrů. Dosahované přenosové rychlosti jsou od stovek B/s po přibližně deset kB/s. Maximální vzdálenost přenosu se liší podle zvolené přenosové rychlosti a kvality kabelu, zpravidla se však jako doporučené maximum udává 15 m. Jde o rozhraní typu peer-to-peer, každé zařízení může zahájit přenos, pomocí jednoho rozhraní je možné připojit pouze jedno zařízení.

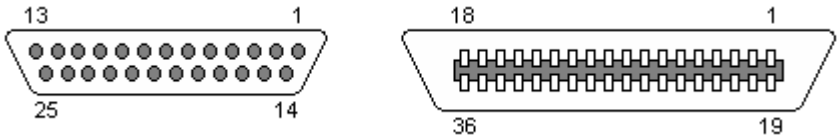


Obrázek 2.1: Konektory sériového rozhraní. Vlevo DE-9, vpravo DB-25. Obrázky převzaty z [2].

2.2.2 LPT

Rozhraní je též známo jako paralelní rozhraní, zkratka LPT znamená *line print terminal*. Vychází z rozhraní Centronics z počátku 70. let 20. století. Rozšířilo se s příchodem IBM PC v 80. letech, u osobních počítačů se zpravidla vyskytoval jeden konektor tohoto rozhraní, po roce 2000 se od tohoto rozhraní upouští, v dnešní době toto rozhraní zpravidla osobní počítače již neobsahují. Jak vyplývá z názvu, jedná se o rozhraní určené především pro připojení tiskáren a jiných zobrazovacích

zařízení. Fyzické rozhraní je paralelní synchronní, s 8 bity dat, zbývající vodiče jsou řídicí a stavové, na straně počítače je zpravidla tvořeno konektorem DB-25 typu samice, na straně koncového zařízení (tiskárny) bývá použito 36 pinového konektoru Centronics. Softwarové rozhraní je velice jednoduché, obsahuje pouze tři 8 bitové registry (datový, stavový a řídicí). Přenosová rychlost je 1,5 KB/s. Pomocí rozhraní je možné připojit pouze jedno zařízení. Přestože je za určitých okolností možný obousměrný provoz, je rozhraní používáno téměř výhradně pro přenos dat z počítače do periferního zařízení.



Obrázek 2.2: Konektory paralelního rozhraní. Vlevo DB-25, vpravo Centronics36. Obrázky převzaty z [2].

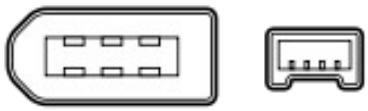
2.2.3 IrDA

Jde o bezdrátové rozhraní definované sdružením IrDA (Infra-red Data Association). Rozhraní bylo definováno v roce 1993. V druhé polovině 90. let se těšilo poměrně velké oblibě, protože umožňovalo připojit zařízení (zejména mobilní telefony, PDA a podobně) za běhu počítače, postupně však bylo téměř vytlačeno rozhraním USB a Bluetooth, dnes je toto rozhraní spíše okrajovou záležitostí pro specifické oblasti použití. Nevýhodou rozhraní je nutnost přímé viditelnosti mezi vysílačem a přijímačem. Na fyzické úrovni rozhraní využívá k přenosu infračervené světlo o vlnové délce 875nm, data jsou přenášena sériově, asynchronně. Na softwarové úrovni je nejčastěji implementováno jako virtuální sériový (COM) port, proto se o IrDA často mluví jako o bezdrátovém sériovém portu. Přenosové rychlosti jsou 9600 b/s až 16 Mb/s. Každé IrDA zařízení musí implementovat rychlost 9600 b/s, která jako výchozí slouží k detekci připojených zařízení a ke konfiguraci následných přenosů. Maximální vzdálenost se udává 1 m s vyosením přijímače a vysílače o úhel 15°, pro nízkopříkonová zařízení je maximální vzdálenost menší.

2.2.4 FireWire

První specifikace rozhraní FireWire (též známá pod označením IEEE 1394) byla vydána v roce 1995. V počátcích byl vydavateli standardu požadován poměrně velký licenční poplatek, což byl jeden z hlavních důvodů, proč se rozhraní nerozšířilo tak masově, jako rozhraní USB, jak je to s licenčními poplatky nyní se mi nepodařilo zjistit. Jde o rozhraní, které se používá a používat se určitě ještě několik dalších let bude, ale rozhraní není standardním vybavením nových počítačů, které by se vyskytovalo u všech nových kusů, přestože je v některých ohledech lepší než „konkurenční“ USB. Používá se zejména v aplikacích, kde je kladen větší důraz na přenosovou rychlost, typicky jde o datová úložiště, videokamery a podobně, přestože rozhraní umožňuje připojit různé druhy zařízení. Na fyzické úrovni se jedná o sériovou sběrnici se dvěma diferenciálními páry vodičů pro data a dvojicí vodičů pro napájení, která podle specifikace může napájet zařízení výkonem až 45 W, v osobních počítačích však tato vlastnost zpravidla nebývá implementována a zařízení je tak třeba napájet nezávislým zdrojem napětí. Sběrnice umožňuje připojit k počítači až 63 zařízení pomocí jednoho konektoru, umožňuje připojení a odpojení zařízení za běhu počítače. Na softwarové úrovni

bývá rozhraní mezi jádrem počítače a rozhraním zpravidla řešeno pomocí OHCI standardu. Mapování paměťového prostoru rozhraní a připojených zařízení je realizováno v hardwaru. OHCI standard definuje softwarové rozhraní na úrovni registrů. Výhodou rozhraní je možnost přímé komunikace zařízení mezi sebou bez potřeby řídicího prvku na sběrnici. Maximální přenosová rychlost původní specifikace je 400 Mb/s, revize standardu z roku 2008 definuje maximální přenosovou rychlost až 3,2 Gb/s. Délka fyzického spoje mezi dvěma zařízeními je omezena na 4,5 m.

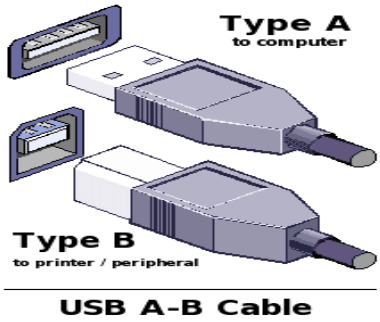


Obrázek 2.3: Konektory FireWire. Vlevo 6 pinový, vpravo 4 pinový.

Obrázek převzat z [3].

2.2.5 USB

První specifikace rozhraní USB byla schválena v roce 1996. Rozhraní je charakteristické tím, že velká část inteligence v rozhraní je umístěna v počítači, samotné zařízení může být velice jednoduché. Jde o rozhraní, které je uživatelsky velice přívětivé, umožňuje připojovat zařízení za běhu počítače, konfigurace periferního zařízení probíhá automaticky, navíc umožňuje napájet energeticky nenáročná zařízení (do 2,5W) přímo ze sběrnice. Výše zmíněné vlastnosti byly předpokladem pro masové rozšíření. Rozšíření dosáhlo takové míry, že dnes nejen každý nový osobní počítač, ale i mnohé domácí spotřebiče, automobily a jiná zařízení, jsou tímto rozhraním vybaveny. Na hardwarové úrovni se jedná o sériovou sběrnici s jedním diferenciálním párem vodičů pro data a párem vodičů pro napájení. Sběrnice umožňuje připojit až 127 zařízení pomocí jednoho konektoru v počítači, na sběrnici je vždy jeden prvek řídicí, ostatní zařízení se tomuto prvku musí podřídit. Na softwarové úrovni existuje více standardů a je třeba rozlišovat, na které úrovni sběrnice je myšleno. Mezi jádrem počítače a řídicím prvkem je jiné softwarové rozhraní než mezi částí rozhraní na straně periferního zařízení a samotnou aplikací periferního zařízení. Softwarových rozhraní řídicího prvku vzniklo postupně více, detailně budou popsány dále, v principu však vždy jde o namapování registrů řídicího prvku do adresového prostoru procesoru. Maximální přenosová rychlost verze 1.0 (1996) a 1.1(1998) je 12 Mb/s, revize 2.0 z roku 2001 zvyšuje maximální rychlost na 480 Mb/s. Poslední verze z roku 2008 definuje maximální rychlost 5 Gb/s. Maximální délka rozhraní mezi dvěma zařízeními je 3 metry.



Obrázek 2.4: Základní typy USB konektorů.

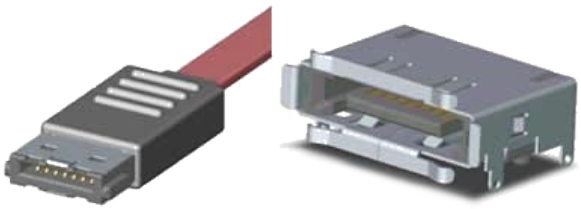
Obrázek převzat z [4].

2.2.6 Bluetooth

První specifikace rozhraní byla vytvořena v roce 1994, standardizována potom v roce 1998. Výrazněji se však rozšířily až pozdější revize, zejména revize 1.1 z roku 2002, respektive 1.2 z roku 2005. Jde o velmi rozšířené rozhraní zejména v oblasti mobilních zařízení a spotřební elektroniky, kde je rozhraní využíváno zejména pro přenos hlasu mezi mobilním telefonem a handsfree zařízením, pro přenos dat mezi počítači je rozhraní využíváno zřídka. Na fyzické úrovni se jedná o bezdrátové rozhraní, které využívá rádiových vln ve frekvenčním pásmu kolem 2,4 GHz. Pomocí jednoho adaptéru je možné připojit více zařízení. Na softwarové úrovni existuje více rozhraní, nejrozšířenějším je HCI (host controller interface), které podobně jako u USB mapuje registry adaptéru do paměťového prostoru procesoru. Přenosová rychlost verze 1.2 je 1 Mb/s, specifikace verze 2.0 z roku 2004 umožňuje až 3 Mb/s. Poslední verze 3.0 umožňuje teoreticky až 24 Mb/s. Dosah rozhraní se liší podle třídy zařízení a podle normy je odstupňován dle výkonu vysílače na 1, 10 a 100 m.

2.2.7 eSATA

Rozhraní bylo standardizováno v roce 2004, jde o variantu rozhraní SATA, umožňující díky upřesnění některých pasáží standardu SATA, připojování vnějších periferních zařízení k osobnímu počítači. V současné době (konec roku 2009) je tímto rozhraním vybavena většina nových osobních počítačů. Rozhraní umožňuje připojit k jednomu konektoru pouze jedno zařízení typu datové úložiště přímo nebo až 15 zařízení s využitím rozbočovačů. Rozhraní se omezuje pouze na zařízení typu datové úložiště, nejdená se tedy o univerzální rozhraní, výhodou však je velká přenosová rychlost. Na fyzické úrovni jde o sériové rozhraní se dvěma diferenciálními páry vodičů pro přenos dat, v rozhraní je dále přítomný signalizační vodič a zemnící vodiče. Na softwarové úrovni je nejčastější rozhraní AHCI, které definuje rozhraní na úrovni registrů a definuje způsob jejich mapování do paměti procesoru. Rozhraní umožňuje teoretickou přenosovou rychlost 3 Gb/s, maximální délka rozhraní je omezena na 2 m.



Obrázek 2.5: Konektor eSATA. Obrázek převzat z [5].

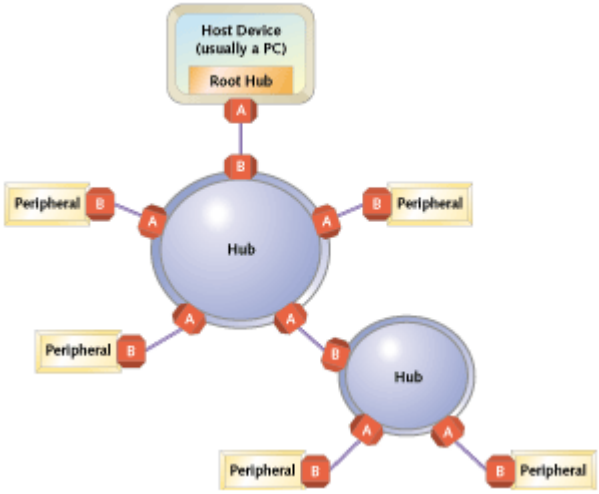
2.3 Detailní popis sběrnice USB

Zkratka USB vychází z anglického spojení *universal serial* bus, neboli univerzální sériová sběrnice. Jde o standard, který oproti předchozímu stavu způsobil v oblasti rozhraní periferních zařízení počítačů malou revoluci. Pod pojmem USB je zahrnuto jak fyzické rozhraní, tak komunikační protokol. Díky výhodám, které USB poskytuje, došlo k jeho masovému rozšíření a v dnešní době je

využíván nejen v počítačích, ale i ve spotřební elektronice a nezřídka i v domácích spotřebičích, automobilech a v řadě dalších průmyslových aplikací.

Mezi klíčové výhody USB patří jednotný konektor v počítači pro celou škálu periferních zařízení (proto je ve zkratce slovo universal), možnost připojit více zařízení pomocí jednoho konektoru počítače, připojení zařízení za běhu počítače a systému, automatická konfigurace zařízení po připojení, vysoká propustnost rozhraní (s ohledem na dobu vzniku), snadná obsluha, možnost napájení zařízení ze sběrnice, rychlý a tím i levný návrh zařízení a obsluhujícího systému, možnost sdružit více funkcí do jednoho zařízení (např. tiskárna, skener a fax v jednom) a další výhody. Některé výše zmíněné výhody budou diskutovány dále.

Rozhraní USB je typu Master/Slave. Zařízení využívající USB rozhraní lze rozdělit do třech skupin: USB hostitel (USB host), USB rozbočovač (USB hub) a koncové USB zařízení (USB device). Hostitel představuje řídicí člen sběrnice, který spravuje veškerá připojená zařízení. Na sběrnici se vyskytuje vždy právě jeden hostitel. Pokud má počítač více konektorů pro připojení USB zařízení, pak obsahuje více hostitelů, existuje tak více nezávislých USB sběrnic v rámci jednoho počítače. Sběrnice vytváří topologii vrstveného stromu (dle [6] jde o topologii vrstvené hvězdy). Hostitel, někdy též nazývaný kořenový rozbočovač, tvoří kořen stromu. Koncová zařízení nebo rozbočovač, ke kterému není žádné další zařízení připojeno, tvoří listy stromu. V rámci jedné sběrnice USB se může vyskytovat až 127 současně připojených zařízení v sedmi úrovních, přičemž 0. úroveň představuje hostitel, rozbočovač se může vyskytovat nanejvýš v 5. úrovni a 6. úroveň může představovat pouze koncové zařízení.

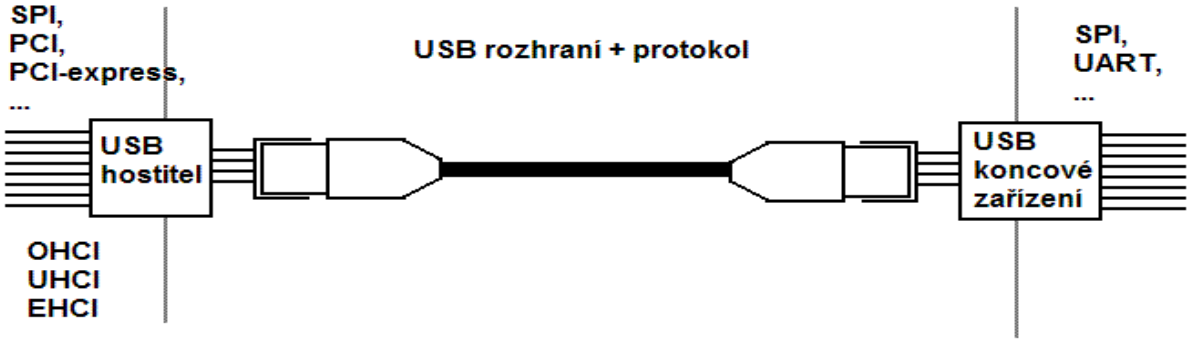


Obrázek 2.6: Topologie sběrnice USB. Obrázek převzat z [7].

Všechny konektory jsou navrženy tak, aby při zasunování konektoru došlo nejprve ke spojení vodičů napájení a teprve potom ke spojení vodičů pro přenos dat. Tím je zabezpečena inicializace připojovaného zařízení, resp. jeho komunikační části, před spojením datových vodičů a je tak zaručena bezproblémová komunikace na sběrnici, přestože je možné zařízení připojovat za běhu počítače.

Pro srozumitelnost dalšího textu je potřeba rozlišit různé úrovně rozhraní. Jednak je možné řešit fyzickou úroveň rozhraní, jednak softwarovou úroveň. V souvislosti se sběrnicí USB je třeba rozlišovat tři části rozhraní, rozhraní hostitele, rozhraní sběrnice USB a rozhraní koncového zařízení vzhledem k aplikaci zařízení. *Rozhraní hostitele* k nadřazenému systému, typicky ve vztahu

k procesoru osobního počítače, je na softwarové úrovni řešeno několika standardy, tyto standardy popisují rozhraní hostitele na úrovni registrů a způsob mapování těchto registrů do adresového prostoru procesoru a budou popsány dále. Na fyzické úrovni je hostitele k nadřazenému zařízení možno připojit například pomocí rozhraní SPI, PCI, PCI-express a další průmyslové sběrnice, případně je možná integrace USB hostitele přímo do mikroprocesoru, což bývá často případ vestavěných zařízení. *Rozhraní sběrnice USB* je na softwarové i fyzické úrovni pevně určeno standardem USB. *Rozhraní* na straně *koncového zařízení* je k aplikaci zařízení řešeno na softwarové úrovni pomocí zjednodušené sady registrů, na fyzické úrovni je možné využití například rozhraní UART, SPI a další průmyslové sběrnice, poměrně častá je přímá integrace USB rozhraní přímo v mikroprocesoru cílové aplikace. Přehledový obrázek by měl usnadnit orientaci ve zmíněných rozhraních.



Obrázek 2.7: Znázornění částí rozhraní.

2.3.1 Verze standardů USB

O standardizaci protokolu a rozhraní se od počátku stará sdružení *USB-IF* (USB Implementers Forum, Inc.), sdružující nejvýznamnější výrobce IT (HP, Intel, LSI, Microsoft, NEC, ST-Ericsson), viz [8]. Kromě vytváření nových standardů se sdružení USB-IF zabývá certifikací produktů, umožňující opatřit výrobek příslušným logem, a přidělováním identifikačních kódů výrobců USB zařízení.

První standard USB 1.0 byl uveden v roce 1996, v roce 1998 prošel revizí na verzi 1.1. V roce 2001 byl uveden USB 2.0, který přináší zvýšení rychlosti proti předchozím standardům. V době vzniku této práce je poslední verzí standardu verze 3.0 z roku 2008. Do verze 2.0 šlo výhradně o nesměrovaný protokol typu dotaz-odpověď, verze 3.0 přináší výrazné změny, protože umožňuje současnou obousměrnou komunikaci, podrobněji budou změny popsány v příslušné podkapitole. Mezi rysy protokolu patří nízká náročnost samotného protokolu na šířku přenosového pásma a implementační jednoduchost zařízení, protože velká část inteligence je svěřena softwaru obsluhujícího řídicí prvek sběrnice.

Koncové zařízení je na sběrnici jednoznačně identifikováno adresou, která je zařízení přidělena hostitelem na počátku komunikace. Počáteční komunikace se nazývá *enumerace*, během které dojde k výměně série zpráv určujících typ zařízení, způsob komunikace, velikost vyrovnávacích bufferů a podobně. Komunikační jednotkou na linkové vrstvě jsou rámce, na síťové vrsvě pakety. Zařízení aktivně naslouchá provoz na sběrnici, pokud je paket adresován tomuto zařízení, tak jej zařízení přečte nebo uloží do bufferu, zkontroluje správnost dat pomocí CRC obsaženého v paketu, potvrdí příjem a příslušným způsobem zareaguje.

Společným rysem všech USB standardů je způsob kódování dat na fyzickém rozhraní. Jde o bitově orientovaný přenos, tedy každý přenášený bit má svůj význam. Protokol využívá *NRZI kódování* s využitím tzv. *bit-stuffingu* (vkládáním bitů). Ve stručnosti je princip bit-stuffingu takový, že do dat určených k přenesení odesílatel za každou sekvenci šesti logických jedniček vloží navíc bit v logické nule, následně je takto vzniklá posloupnost bitů zakódována NRZI kódováním, kdy logická jednička znamená zachování stávající úrovně a logická nula znamená změnu logické úrovně. Přijímač pracuje obráceně, nejdříve z napěťových úrovní určí posloupnost bitů a každou logickou nulu, která následuje po šesti logických jedničkách, odstraní. Tím je zaručeno, že nejpozději každý sedmý bit dochází ke změně úrovně, což umožňuje snadnou synchronizaci koncového zařízení, které tím pádem nemusí mít tak přesný generátor hodinového signálu.

2.3.1.1 USB 1.0 a 1.1

Verze 1.0 byla implementována pouze u málého počtu zařízení a nedošlo k jejímu výraznému rozšíření, důležitá je zejména z toho důvodu, že definuje dva hlavní typy konektorů, viz Obrázek 2.4. Konektor typu A se užívá vždy směrem k hostiteli, konektor typu B vždy směrem ke koncovému zařízení. Na fyzické úrovni je rozhraní čtyřvodičové, dva vodiče slouží pro napájení zařízení napětím 5V, zbylé dva tvoří diferenciální pár pro přenos dat. Verze 1.1 přináší drobné změny, zejména v komunikaci s USB rozbočovači, jde o první masově rozšířenou a implementovanou verzi USB standardu. Z hlediska protokolu nepřináší verze 1.1 nic nového, z hlediska rychlosti přenosu přináší osminásobné zrychlení z původních 1,5 Mb/s na 12 Mb/s zkrácením doby trvání přenosu jednoho bitu na osminu původní hodnoty. Dále verze 1.1 definuje délku spojení mezi dvěma konci linky na 5 m místo dosavadních 3 m.

Zařízení využívající rychlost přenosu 1,5 Mb/s jsou nazývána *low-speed*, zařízení využívající rychlost přenosu 12 Mb/s se nazývají *full-speed*. Někdy tyto alternativní názvy slouží k identifikaci standardu místo uvádění jeho verze.

2.3.1.2 USB 2.0

Verze 2.0 z roku 2001 je v současné době nejrozšířenějším standardem, jde o rozšíření verze 1.1 při zachování stejného fyzického rozhraní, konektory tak zůstavají stejné. V nové verzi je implementován vysokorychlostní přenos s teoretickou propustností až 480 Mb/s označovaný jako *high-speed*, v praxi je tato propustnost nižší kvůli režii protokolu a prodlevám v sw obsluze. Určitou zajímavístí je, že koncové zařízení vyhovující standardu 2.0 nemusí zmíněný vysokorychlostní přenos vůbec implementovat, to však neplatí pro rozbočovače. Je-li rozbočovač standardu 2.0 připojen k hostiteli standardu 2.0, poté komunikace mezi rozbočovačem a hostitelem probíhá vždy rychlostí 480 Mb/s. Rozbočovač je poté zodpovědný za komunikaci nižší rychlosti mezi rozbočovačem a k němu připojeným koncovým zařízením, které high-speed přenos nevyužívají. Rozdílnou rychlostí přenosu téže informace ke koncovému zařízení přes rozbočovač je ušetřeno přenosové pásmo mezi hostitelem a rozbočovačem.

2.3.1.3 USB 3.0

Prokokol verze 3.0 je v době vzniku této práce nejnovějším protokolem, standard byl schválen v listopadu roku 2008. V současné době se začínají objevovat na trhu první zařízení, která tento protokol implementují, rovněž se objevují první počíače standardně vybavené tímto rozhraním.

Oproti předchozím verzím přináší opět výrazné zrychlení a to až na rychlosti 5 Gb/s (4,8 Gb/s efektivních dat). Tyto extrémně rychlé přenosy se nazývají *super-speed*. Kromě zvýšení propustnosti rozšiřuje možnosti energetické úspornosti zavedením vícestupňového řízení spotřeby.

Z hlediska rozhraní se jedná o velký rozdíl proti předchozím verzím. Specifikace USB 3.0 v podstatě přináší paralelní spojení USB 2.0 a SuperSpeed rozhraní. Přidané rozhraní znamená jiné typy konektorů (viz Obrázek 2.8) a větší počet vodičů v rozhraní, do kterého jsou přidány dva diferenciální páry pro přenos dat, existence samotného super-speed není možná. Vše je navrženo pro co největší kompatibilitu. Všechna koncová zařízení předchozích verzí je možné zasunout do konektorů USB 3.0. Zařízení standardu 3.0 je možné zasunout do konektoru standardu 2.0, avšak komunikace se může omezit na požadavek koncového zařízení na připojení k hostiteli standardu 3.0 nebo je v takovém případě snížena rychlost a připojené zařízení zvolí konfiguraci s maximální rychlostí 480 Mb/s.

SuperSpeed protokol již není typu dotaz-odpověď, ale jedná se o protokol asynchronního přenosu, který je navíc, na rozdíl od předchozích verzí, směrovaný. Jednotlivá data jsou distribuována pouze po cestě mezi zdrojem a cílem, v předchozích verzích docházelo ke sdílení a data byla distribuována všem, přičemž pouze zdroj a cíl konkrétní transakce data produkoval/konzumoval. Dalším významným rozdílem je fakt, že u předchozích verzí mezi sebou nemohla komunikovat dvě koncová zařízení přímo, ale prostřednictvím hostitele. USB 3.0 zavádí možnost komunikace dvou koncových zařízení přímo pomocí speciálního kabelu.

Každé USB 3.0 zařízení musí implementovat i některou z předchozích variant k dodržení kompatibility. To, že zařízení musí implementovat předchozí verzi, neznamená jeho plnou funkčnost na předchozí verzi. Funkčnost může být omezena nebo může pomocí předchozí verze pouze s hostitelem vykomunikovat, že ke své funkci zařízení potřebuje USB 3.0 řadič.

Obrázek 2.8: Konektory standardu USB 3.0. Obrázky převzaty z [9] a [10].

2.3.1.4 On-The-Go a Embedded host

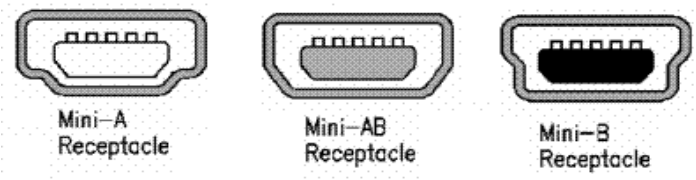
Standard On-The-Go[11] byl vytvořen jako dodatek k USB protokolu verze 2.0 jako reakce na rostoucí požadavky na připojování USB zařízení k jiným výpočetním prostředkům než běžným počítačům (průmyslové aplikace) nebo v případech, kdy byla od výrobku požadována funkcionalita koncového zařízení i potřeba další zařízení autonomně obsluhovat, vždy však pouze jedna z uvedených možností v jeden okamžik. Standard definuje rozšíření, které umožňuje zařízení chovat se za určitých okolností jako hostitel, podporu přímého propojení dvou takovýchto (OTG) zařízení, možnosti úspory energie pro prodloužení doby běhu zařízení na baterie a podobu malých USB

konektorů pro umístění do mobilních zařízení viz Obrázek 2.9. Při přímém propojení dvou zařízení se jedno zařízení chová jako hostitel, druhé jako koncové zařízení.

To, zda se bude OTG zařízení chovat jako koncové zařízení, nebo jako hostitel, je umožněno díky novému konektoru (mini-AB, viz Obrázek 2.9 uprostřed), který obsahuje 5 vodičů, zatímco standardní konektory mají pouze 4 vodiče. Pokud zařízení po připojení detekuje napětí na sběrnici, chová se jako koncové zařízení, pokud ale napětí na sběrnici nedetekuje, začne samo sběrnici napájet, tím je zaručeno, že po připojení OTG zařízení k běžnému hostiteli v počítači se zařízení bude vždy chovat jako koncové zařízení a nedojde tak ke konfliktu, protože hostitel může být na sběrnici právě jeden. Ve chvíli, kdy se k takovému zařízení připojí jiné koncové zařízení, tak toto koncové zařízení uzemní pátý vodič v konektoru a tím jasně určuje, že OTG zařízení bude fungovat v roli hostitele.

Zařízení typu Embedeed host funguje velice podobně jako zařízení typu OTG, jenom s tím rozdílem, že pro funkci hostitele je vyhrazen jiný konektor (mini-A, viz Obrázek 2.9 vlevo) než pro funkci koncového zařízení. U zařízení standardu OTG je přítomný právě jeden společný konektor.

Hostitelé tohoto druhu nemusí podporovat všechny režimy přenosu, všechny standardy rychlosti. Mohou například podporovat právě jedno konkrétní zařízení a žádné jiné. Ze standardů musí podporovat právě ty části, které jsou vyžadovány cílovými koncovými zařízeními, které se k hostiteli budou připojovat, veškeré další části mohou být podporovány volitelně nebo nemusí být podporovány vůbec. V souvislosti s výše uvedeným se takovým hostitelům říká zjednodušený hostitel (anglicky *reduced host*).



Obrázek 2.9: Mini USB konektory (zleva pro EH, OTG, koncové zařízení).

Obrázky převzaty z [12].

2.3.2 Typy přenosu

V rámci protokolu USB se rozlišují čtyři druhy přenosu – řídicí (control), běžný (bulk), vyzývaný (interrupt) a izochronní (isochronous).

Řídicí přenos slouží pro obousměrnou komunikaci mezi zařízením a hositelem. Hostitel pomocí tohoto druhu přenosu zasílá zařízení příkazy a dotazy, zařízení pomocí něj na dotazy odpovídá. Přikladem takové komunikace je například proces enumerace, kdy dochází k zjištění, jakého je zařízení druhu, přidělení adresy a podobně. Tento typ přenosu je přítomný u všech standardů protokolu, podle verze protokolu se může lišit délka paketu.

Běžný přenos využívá přenosovou šířku pásma, která zbyde po odečtení zbylých třech druhů přenosu. Přenos je využíván pro časově nekritické přenosy, protože není garantována šířka pásma ani komunikační zpoždění. Výhodou přenosu je bezpečnost přenášených dat – data jsou zabezpečena pomocí CRC a v případě chyby při přenosu jsou přenášena opakovaně.

Vyzývaný přenos slouží k přenosu dat v pravidelných intervalech, interval je nastaven při procesu enumerace. Anglický název *interrupt* je poněkud nešťastný, protože přenos probíhá pouze po vyzvání hostitelem, zařízení nemá možnost odeslat přerušení (neplatí pro USB 3.0). Slouží k přenosu spíše menších objemů dat, která jsou citlivá na čas, typicky jde o přenos dat informujících o stavu

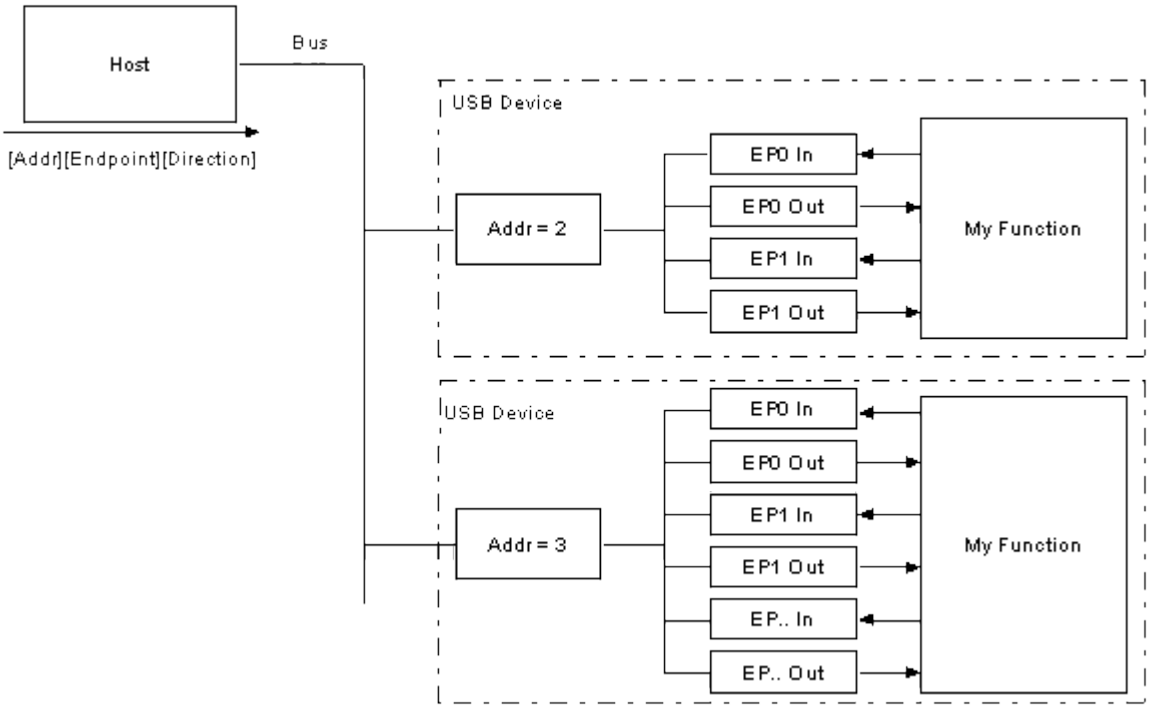
zařízení, například změna stavu linky u modemu. Výhodou přenosu je garantované komunikační zpoždění a detekce chyb při přenosu s případným znovuzasláním dat při příštím vyzvání.

Izochronní přenos se používá pro přenosy real-time informací, kde je vhodnější případný výpadek z poškození dat při přenosu než zpoždění vzniklé znovuzasíláním informací nebo kde znovuzasílání z nějakých důvodů není možné (data nejsou na straně koncového zařízení bufferována). Typicky jde o přenosy audiovizuálních dat, například data z mikrofonu. Výhodou přenosu je garantovaná šířka přenosového pásma a jasně ohraničené komunikační zpoždění. Při přenosu probíhá detekce chyb pomocí CRC, v případě chyby ale data nejsou znovu zasílána. Tento režim přenosu není dostupný u pomalých zařízení typu low-speed.

2.3.3 Adresování na USB sběrnici, koncové body

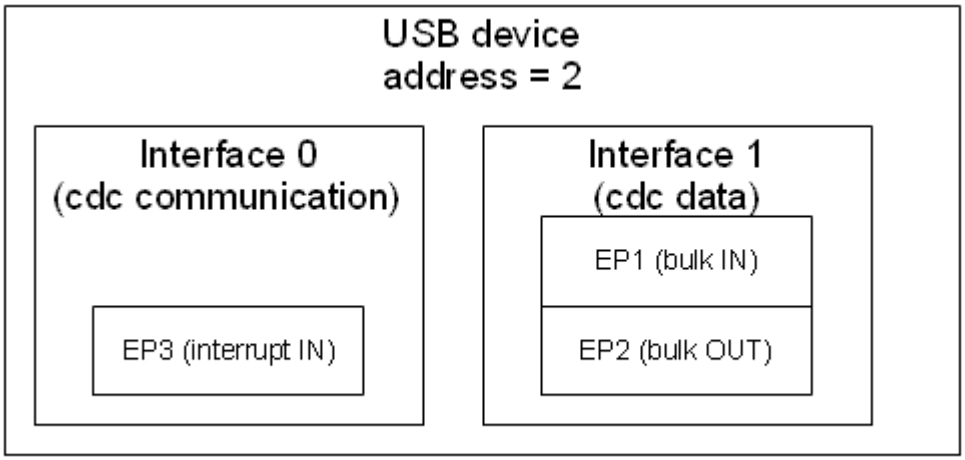
Každé zařízení má po připojení hostitelem přidělenu jednoznačnou adresu na sběrnici. Komunikace mezi hostitelem a zařízením na bázi protokolu probíhá pomocí *rour* (pipes). Jedno zařízení může současně využívat více rour. Roury jsou dvou druhů, jeden druh je pro obousměrné zasílání zpráv (message pipe), ten je vyhrazen pouze řídicím přenosům, druhý druh je pro jednosměrné zasílání proudů (stream pipe), ten je určen pro ostatní druhy přenosů. Roury jsou adresovány adresou zařízení a adresou koncového bodu zařízení. V rámci jednoho zařízení mohou být 4 koncové body pro low-speed zařízení respektive po 16 koncových bodech pro každý směr přenosu u full-speed zařízení. V rámci těchto počtů musí být zahrnut koncový bod 0, který musí být vždy přítomný a který je vždy využíván pro řídicí přenosy. Koncový bod 0 slouží pro konfiguraci zařízení, zjišťování stavu zařízení a jeho nastavování.

Nabízí se analogie s IP protokolem. Adresu zařízení si lze představit jako IP adresu, číslo koncového bodu jako port. Koncový bod je na straně hostitele zpravidla svázán s aplikací běžící nad hostitelem. Princip adresování a využití koncových bodů usnadní Obrázek 2.10, kde USB device představuje jednotlivá fyzická zařízení, EPx znamená číslo koncového bodu a My Function znamená funkci poskytovanou zařízením, například vstup z klávesnice, myši nebo sériový port.



Obrázek 2.10: Princip adresování a koncové body USB sběrnice. Obrázek převzat z [13].

Koncové body jsou často sdružovány do skupin nazývaných rozhraní (interface). Ve skupině jsou sdruženy zpravidla koncové body téhož druhu, například běžné zařízení třídy CDC používá většinou (přesný formát není předepsaný, může se lišit dle konkrétního použití, protože třída CDC pokrývá široké spektrum zařízení) dvě rozhraní, z nichž jedno je určeno pro signalizování stavu a využívá vyzývaný přenos s jedním koncovým bodem, druhé rozhraní obsahuje dva koncové body, které slouží pro přenos dat, k přenosu dat je používán běžný přenos. Typický příklad konfigurace zařízení typu CDC (např. modem nebo USB-RS232 převodník) viz Obrázek 2.11.



Obrázek 2.11: Konfigurace zařízení třídy CDC.

2.3.4 Deskriptory, proces enumerace

Po připojení zařízení na sběrnici má zařízení adresu 0 a nemůže se účastnit přenosu mimo řídicí přenos. Hostitel detekuje připojení nového zařízení pomocí změny napěťových úrovní na datových vodičích, podle toho, na kterém datovém vodiči došlo ke změně napěťové úrovně, je určeno, jakého je zařízení druhu, jestli low-speed, nebo full-speed (high-speed zařízení se detekuje jako full-speed). Jakmile hostitel detekuje nové zařízení, resetuje sběrnici (oba datové vodiče tvořící diferenciální pár jsou uvedeny do stavu logické 0) a po uvolnění sběrnice ze stavu resetu zahájí proces tzv. *enumerace*. Během tohoto procesu dochází mezi hostitelem a zařízením k výměně informací, které popisují vlastnosti zařízení a dochází k přidělení prostředků pro komunikaci a napájení. Proces enumerace je detailně popsán v podkapitole 4.2.1. Detailní popis významu jednotlivých deskriptorů viz [14].

Informace vyměňované během procesu enumerace se nazývají *deskriptory*. Kromě deskriptorů během enumerace dochází k zasílání příkazů. V deskriptorech jsou uloženy informace o výrobci zařízení, slovního názvu zařízení, verzi standardu USB daného zařízení, požadovaný odběr proudu ze sběrnice (do dokončení procesu enumerace je proud odebíraný zařízením omezen na 100mA, po dokončení může být až 500mA, u USB 3.0 dokonce více). Deskriptory mohou definovat více konfigurací zařízení, hostitel si vybere jednu z nich. Konfigurací se rozumí různé režimy napájení, různé kombinace rozhraní (funkcí) a koncových bodů a jejich vlastnosti jako typ a směr přenosu dat, maximální délka paketu, požadovaná frekvence vyzývání (tím je definována požadovaná šířka přenosového pásma). Deskriptorů je více druhů, dohromady tvoří hierarchii nastavení.

Deskriptor zařízení obsahuje kód výrobce a kód zařízení, určuje délku paketu řídicího koncového bodu, počet konfigurací a může obsahovat informace o třídě a podtřídě zařízení, případně protokol pro komunikaci se zařízením na aplikační úrovni. Na základě těchto informací může hostitel vynutit načtení příslušného ovladače zařízení. Tento deskriptor je vždy pro zařízení právě jeden.

Deskriptor konfigurace obsahuje informace o možnostech napájení, dále obsahuje informace o rozhraních a koncových bodech dané konfigurace. Mezi možnosti napájení patří informace určující způsob napájení, zda bude zařízení napájeno ze sběrnice, nebo jestli má vlastní zdroj napájení, a požadovaný odběr proudu ze sběrnice. Informace o rozhraních a koncových bodech je předána pomocí deskriptorů rozhraní a deskriptorů koncových bodů. Konfiguračních deskriptorů může být obecně více (zpravidla je však pouze jeden), pokud jich je více, hostitel si vybere jednu konfiguraci, například podle možností napájení. Možným příkladem volby jiné konfigurace je stav, kdy právě připojené zařízení obsahuje více konfigurací lišících se požadovaným proudovým odběrem zařízení. Celkový proud odebíraný všemi zařízeními na sběrnici je omezen, proto hostitel může zvolit konfiguraci s nižším požadavkem na napájení, aby nedošlo k překročení této meze. Konfigurační deskriptor neobsahuje žádnou informaci o druhu zařízení.

Deskriptor rozhraní určuje třídu a podtřídu rozhraní, komunikační protokol na aplikační úrovni a počet koncových bodů daného rozhraní. Deskriptorů rozhraní může být v rámci jedné konfigurace více, může obsahovat více koncových bodů. Pokud ještě nebyl načten ovladač zařízení, tak informace obsažená v deskriptoru rozhraní dodává potřebnou informaci pro jeho načtení.

Deskriptor koncového bodu udává směr přenosu mezi daným koncovým bodem a hostitelem, maximální délku přenášeného paketu a druh přenosu, dále ještě požadavek na četnost vyzývání koncového bodu hostitelem, spolu s délkou přenášeného paketu tak určuje požadovanou šířku přenosového pásma. Informace o koncových bodech jsou pro hostitele klíčové z důvodu mapování vyrovnávacích bufferů jednotlivých koncových bodů ve vnitřní paměti.

Textové deskriptory mohou volitelně obsahovat textové informace, které slouží ke snadné interpretaci člověkem. Pomocí těchto deskriptorů jsou předávány informace o výrobci, názvu výrobku, podporovaných jazycích a podobně. Tyto informace mohou sloužit například pro snadnou identifikaci zařízení uživatelem ve správci hardwaru operačního systému, pro člověka je jistě čitelnější, je-li jako výrobce zařízení uveden například Atmel místo 0x03EB.

Kromě výše uvedených deskriptorů se mohou vyskytovat ještě další v závislosti na druhu zařízení. Pro příklad takového deskriptoru uvedu třeba deskriptor popisující rozdělení jednotlivých položek paketu zprávy počítačové myši, který určuje počet dostupných tlačítek a podobně.

2.3.5 Koncová USB zařízení

Koncová zařízení mohou být jednoduchá nebo složená. Jednoduchá zařízení jsou pouze jednoúčelová, složená zařízení sdružují více funkcí v jednom fyzickém zařízení. Funkce zařízení je určena pomocí dekriptoru rozhraní. V rámci jedné konfigurace je možné definovat více rozhraní, v rámci rozhraní více koncových bodů. všechny funkce mohou být téhož typu (třídy zařízení) nebo může být třída zařízení individuálně určená podle rozhraní. Příkladem zařízení se sjednocenými rozhraními může být USB modul setu bezdrátové klávesnice a myši, kde pomocí jednoho adaptéru je možné ovládat jak myš, tak klávesnici zároveň. Příkladem zařízení, kde je každá funkce jiného typu může být mobilní telefon, sdružující sériové zařízení (modem) a velkokapacitní paměťové zařízení.

Moderní operační systémy standardní zařízení podporují automaticky a není potřeba instalovat ovladače. Problém ale může nastat se složenými zařízeními, kde nemusí být podpora ze strany operačního systému zajištěna. Konkrétně tento problém určitě platí pro kombinaci sériového a velkokapacitního zařízení u operačního systému Windows XP. Jediným řešením jak tuto situaci řešit je instalace specifického ovladače pro konkrétní zařízení.

2.3.6 Softwarové rozhraní USB hostitele

USB host spravuje provoz na sběrnici, komunikuje s připojenými zařízeními a pomocí dalšího rozhraní komunikuje s nadřazeným systémem. USB host přiděluje adresy připojeným zařízením a spravuje komunikační prostředky. Z hlediska nadřazeného systému má toto řešení velkou výhodu kvůli správě zařízení připojených k USB sběrnici, která je velmi jednoduchá na implementaci, protože nadřazeným systémem je ovládáno jediné fyzické zařízení, je třeba obsluhovat jeden zdroj přerušení, spravovat jeden vstupně výstupní adresový prostor.

Softwarových rozhraní USB hostů je více druhů. Mezi standardizované typy patří UHCI, OHCI a EHCI hostitelé. V principu jde o standardy rozhraní na úrovni registrů a jejich adresace v adresovém prostoru procesoru nadřazeného systému. Tyto standardy jsou především používány v oblasti osobních počítačů, ve vestavěných systémech je většinou situace výrobce od výrobce různá.

2.3.6.1 UHCI

UHCI je zkratkou za *universal host controller interface*, česky rozrhaní univerzálního USB hostitele. Standard byl vyvinut firmou Intel. Jde o rozhraní pro hostitele se dvěma kořenovými rozbočovači standardu USB 1.1. Softwarové rozhraní obsahuje osm registrů. Konkrétně jde o 16 bitové registry pro příkaz, stav, přerušení, číslo rámce a dva stavové/řídicí registry kořenových rozbočovačů, dále 32 bitový registr adresy v paměti a 8 bitový registr pro synchronizaci sběrnice.

2.3.6.2 OHCI

Jde o zkratku *open host controller interface* neboli otevřené (volně dostupné) rozhraní USB hostitele. Pod standardem jsou podepsány firmy Compaq, Microsoft a National Semiconductor. Standard vyhovuje požadavkům USB 1.1. Standard je primárně určen pro architekturu x86, všechny registry jsou 32 bitové. Standard definuje čtyři druhy registrů, konkrétně jde o registry řídicí a stavové, registry ukazatelů do paměti, registry čítače rámců a registry kořenového rozbočovače. Řídicí a stavové registry slouží k obsluze přerušení a základních vlastností. Registry ukazatelů do paměti slouží k určení výlučných oblastí paměti nadřazeného systému pomocí kterých si hostitel a systém předávají data.

2.3.6.3 EHCI

Zkratka vychází z anglického sousloví *enhanced host controller interface* neboli rozhraní vylepšeného hostitele. Tento standard definuje rozhraní USB hostitele a nadřazeného systému na úrovni registrů, vychází ze standardu OHCI. Proti předchozím řešením přináší rozšíření v podobě podpory full-speed zařízení, rozšířené možnosti správy napájení a počítá s rozšiřitelností do budoucnosti, proto podporuje i 64bitové adresování. Tento standard plně pokrývá novinky protokolu USB 2.0. Standard je podepsán všemi významnými hráči na trhu IT (konkrétně Intel, Compaq, NEC, Lucent a Microsoft), proto také v současné době patří mezi nejrozšířenější standard softwarového rozhraní USB hostitelů. Sjednocení standardu má výhodu pro implementátory operačních systémů, protože stačí pouze jeden ovladač pro všechna hostitelská zařízení pro standard USB 2.0.

Registry definované standardem jsou rozděleny do třech skupin. První skupina registrů je pro PCI rozhraní hostitele, druhá skupina slouží pro získání informací o hostiteli nadřazeným systémem a třetí skupina slouží pro obsluhu samotného USB hostitele. Všechny registry obsluhy hostitele jsou 32 bitové.

2.3.7 Vybrané USB řadiče pro vestavěné systémy

V této podkapitole budou představeny možnosti implementace USB rozhraní ve vestavěných systémech. Mezi tyto možnosti patří zejména dedikované čipy poskytující USB rozhraní pro aplikaci jako USB zařízení, hostitel nebo obojí, dále budou jako další možnost zmíněny vybrané mikrokontroléry obsahující USB rozhraní.

Výhodou dedikovaných čipů obecně je odlehčení procesorové zátěže samotné aplikace spojené s řízením USB rozhraní, rovněž jsou výrazně sníženy nároky na zkušenosti a schopnosti programátora, který tak nemusí znát a zvládat detaily spojené s obsluhou USB rozhraní. Nevýhodou dedikovaných čipů je větší složitost hardwaru, nutný větší počet součástek, který zvyšuje pravděpodobnost vzniku poruch při provozu zařízení. Rovněž toto řešení zvyšuje jednotkovou cenu, která je u sériově vyráběných zařízení zásadní. Diskutovány budou výhody a nevýhody vybraných konkrétních řešení.

2.3.7.1 Vinculum

Vinculum je název pro čip VNC1L od firmy FTDI, jde o čip obsahující dva USB řadiče s možností funkce jako USB hositel nebo koncové zařízení. Čip dále obsahuje rozhraní UART, SPI a FIFO pro připojení k nadřazenému systému, čímž umožňuje snadno rozšířit vestavěné zařízení o rozhraní USB.

Na softwarové úrovni se čip ovládá výhradně pomocí příkazového jazyka. Příkazový jazyk je buď v ASCII formě pro snadné ovládání pomocí terminálu, nebo v hexadecimální formě pro snížení objemu přenášených dat při komunikaci ve vestavěné aplikaci. Podporovaná sada příkazů je zaměřena především na práci se soubory na přenosných paměťových médiích. Čip podporuje jak low-speed, tak full-speed přenosy. Standardně je (dle informací výrobce) čip dodáván bez firmwaru, který je možno do prázdného čipu nahrát pouze pomocí UART rozhraní nebo v dedikovaném programátoru. Pokud již je v čipu nějaký firmware nahrán, je možné nahrát jiný i z USB flash disku.

Mezi výhody patří velmi snadné použití a dostupnost i v České republice. Existují dokonce i hotové osazené moduly do DIP patic. Výrobce dodává 6 základních firmware, které pokrývají většinu běžně používaných zařízení jako flash disky, klávesnice, myši, převodníky na RS232 a podobně. Výrobce dodává program pro OS Windows, ve kterém je možné nastavit některé výchozí parametry firmwaru jako výchozí parametry rozhraní UART, možnost zamknout čip proti přeprogramování a podobně. Výhodou je možnost snadno začlenit čip do stávajícího zařízení s malým zásahem do hw.

Mezi nevýhody patří poměrně vysoká cena, která se pohybuje kolem 260 Kč za samotný integrovaný obvod a 1000 Kč za hotový modul. Rovněž je problematické velmi netypické SPI rozhraní, které využívá 13 cyklů hodin k přenesení jednoho byte dat a signál CS používá opačnou polaritu, než je obvyklé (je aktivní v log 1, standardní je invertované chování). Rovněž nemožnost programovat prázdný čip pomocí jiného než UART rozhraní může být problém, protože čip musí být naprogramován mimo aplikaci nebo musí být deska plošného spoje rozšířena o součástky umožňující prvotní naprogramování, přestože toto rozhraní není v konkrétní aplikaci využito. Hlavní nevýhodou, znemožňující nasazení mimo prototypování nebo amatérské aplikace, je dle mého názoru nemožnost implementace vlastního firmware nebo alespoň několika uživatelských příkazů. Při použití jiného než podporovaného zařízení je sice komunikace v režimu hostitele se zařízením možná, ale přenáší výrazně větší zátěž pro nadřazený systém než u podporovaných druhů zařízení, rovněž sada příkazů by mohla být obsáhlejší, některé běžné příkazy pro práci s paměťovými médii je třeba složitě obcházet. Posledně zmíněné nedostatky spolu úzce souvisí, pokud by byla sada příkazů o něco bohatší, dalo by se prominout absenci možnosti vytvořit vlastní firmware, zejména schází příkazy pro čtení nebo zápis sektoru připojených paměťových zařízení, což je v průmyslových aplikacích častý požadavek.

2.3.7.2 USBwiz

USBwiz je čipset od firmy GHI Electronics, který prostřednictvím čipu Philips ISP1160 poskytuje služby USB hostitele, služby USB koncového zařízení nejsou podporována. Hostitelský čip Philips je z hlediska interního ovládání odvozen od standardu OHCI, podporuje standard USB 2.0, podporuje rychlosti low-speed a full-speed. Čip USBwiz poskytuje rozhraní UART, SPI a I2C pro připojení k nadřazenému systému. Na softwarové úrovni se čip ovládá pomocí příkazového jazyka. Sada příkazů je zaměřena zejména na práci s vyměnitelnými datovými úložišti. Přeprogramování firmwaru na novější verzi je možné z připojených datových úložišť, kromě USB flash disků to může být i paměťová karta SD nebo MMC.

Mezi výhody patří snadnost nasazení, velmi dobře propracovaná sada příkazů, při práci s datovými úložišti jsem nenarazil na příkaz, který by mi chyběl. Existují i kompletní moduly, které stačí pouze připojit k nadřazenému systému. Další výhodou je podpora více logických svazků v jednom fyzickém disku. Jedině uvítat lze rovněž standardní chování na rozhraní SPI.

Mezi nevýhody patří jednoznačně cena a nedostupnost čipu v České republice. Cena samotného čipsetu je v přepočtu asi 550 Kč, k tomu je třeba připočítat cenu hostitele ISP1160, která je v přepočtu dalších asi 100 Kč, výsledné řešení je potom nejen poměrně drahé, ale zabírá velkou plochu. Na druhou stranu cena hotového modulu je přijatelných 1100 Kč, přičemž tento modul kromě dvou USB konektorů obsahuje i slot na paměťové karty. Jako nevýhoda se rovněž může jevit nižší maximální frekvence rozhraní SPI, která je omezena na 7 MHz. Možnost programování vlastního firmwaru sice není, sada příkazů je však dostatečně obsáhlá, aby to nečinilo velké potíže při případném nasazení v praxi.

2.3.7.3 MAX3421E

MAX3421E nebo odvozená varianta je čip firmy Maxim, který umožňuje rozšířit aplikaci o rozhraní USB. Čip se k aplikaci připojuje pomocí SPI, další rozhraní čip nemá. Může fungovat v režimu USB koncového zařízení nebo USB hostitele, nejde však o standard On-The-Go, volba se provádí manuálně v registrech. Podporuje standard USB 2.0 rychlostí low-speed a high-speed. Na softwarové úrovni neposkytuje žádné aplikační rozhraní, pouze sadu registrů.

Mezi výhody patří nízká cena (v přepočtu asi 70 Kč) a malé rozměry (5x5 mm). Výhodou je velmi rychlé SPI rozhraní, které je možno taktovat až na 26 MHz, rozhraní tedy při přenosech netvoří úzké hrdlo.

Jako nevýhoda může být považováno rozhraní, kdy je potřeba programovat veškeré obslužné rutiny jako adresování, enumeraci vlastními silami. Druhou stranou téže mince je výhoda jednotného přístupu ke všem funkcím, možnost mít veškerou komunikaci plně pod kontrolou.

2.3.7.4 Mikrokontroléry

Poslední explicitně uvedenou podkapitolou jsou mikrokontroléry vybavené USB rozhraním. Mám osobní zkušenost s mikrokontroléry Atmel řady AT90USB z rodiny 8 bitových AVR a AT32UCA a AT32UCB z rodiny 32 bitových AVR a Microchip PIC z rodiny PIC24FJ256GB1, všechny s podporou standardu USB On-The-Go.

Výhodou zmíněných mikrokontrolérů je dostupnost v České republice, nízká cena, existence vývojových kitů a vývojových prostředí. Cena čipu řady AT90USB je přibližně 250 Kč, demonstrační vývojový kit stojí asi 1000 Kč, cena kitu pro PIC je asi 1750 Kč. Čipy AT32UCB stojí kolem 250 Kč, vývojový kit EVK1100 pro AT32UCA stojí asi 2500 Kč, přičemž tento kit obsahuje celou řadu rozhraní včetně ethernetu a umožňuje vývoj náročných profesionálních aplikací. Obecnou výhodou mikrokontrolérů je možnost programování vlastní aplikace, zároveň odpadají starosti s rozhraním USB řadiče, registry USB rozhraní jsou součástí adresového prostoru procesoru. S využitím mikrokontrolérů je možné vytvořit jak aplikaci ušitou přímo na míru konkrétním požadavkům, tak vysoce univerzální prostředí, které je možné kdykoliv přizpůsobit, jen je třeba brát v úvahu dále uvedené omezení.

Nevýhodou může za určitých okolností být omezená funkčnost takového zařízení v roli hostitele, kdy je zpravidla možné implementovat pouze redukovaného hostitele, tedy lze obsluhovat pouze jedno přímo připojené zařízení. Dodávaný software rovněž nemusí obsahovat podporu všech zařízení, ale pouze některých masově rozšířených.

Výše uvedená existující řešení se nepokoušejí obsáhnout celý trh, což je vzhledem k míře rozšíření USB asi nemožné. Spíše šlo o snahu ukázat různé přístupy k řešení problematiky rozhraní

USB ve vestavěných systémech a upozornit na výhody a nevýhody jednotlivých řešení. První dvě řešení (Vinculum a USBwiz) jsou přístupem připojením hotového „univerzálního“ řešení s jednoduchým ovládáním, ale i s omezeními, se kterými je třeba počítat a které nelze dodatečně změnit. Řešení s využitím řadiče Maxim je univerzální, levné na kusovou cenu, ale náročné na implementaci obslužného softwaru. Řešení s mikrokontroléry je ve vestavěných systémech dle mého názoru vhodným průsečíkem předchozích řešení. Umožňuje snadné vytvoření vhodného řešení při zachování flexibility pro přizpůsobení se konkrétním požadavkům.

3 Definice úkolu

Úkolem mé práce je vytvořit systém, který umožní připojení sériového zařízení (modem, telefon GSM) pomocí USB rozhraní mikrokontroléru PIC. Před započetím vlasntího návrhu bylo nutné nastudovat USB protokol, způsob komunikace na sběrnici, procesy, které následují po připojení koncového zařízení (např. modem) k USB sběrnici. Dále bylo třeba prostudovat existující řešení pro získání přehledu, jakým způsobem je možné tuto problematiku řešit.

Způsobů řešení je více a liší se v míře přenositelnosti a znovupoužitelnosti. Je možné vytvořit USB hostitele přímo na míru konkrétní aplikaci, stejně jako je možné vytvořit univerzálního hostitele s možností připojit a ovládat jakékoliv zařízení. Rovněž je možné využít existujících fyzických řadičů, které toto řeší autonomně.

Řešení vytvořit USB hostitele přímo na míru jednomu konkrétnímu zařízení je krajně nešťastným řešením, zejména pokud ono konkrétní zařízení nevyhovuje zcela USB specifikaci. Malá odchylka od standardu může znamenat, že jiné zařízení téhož druhu od jiného výrobce nemusí s takovým hostitelem fungovat.

Využití samostatných autonomních USB řadičů je vhodné pro rychlé prototypování, pro rychlé rozšíření jednoho pokusného zařízení, pro některé druhy aplikací dokonce může být i vhodným koncovým řešením. Hlavní nevýhodu takového řešení považuji ve fixaci na jednoho výrobce, na jeden konkrétní výrobek bez možnosti ovlivnit funkčnost rozhraní. Migrace na jiný výrobek s sebou nese nutnost změny celého rozhraní, protože jiný řadič využívá jiné syntaxe i sémantiky příkazového jazyka.

Vytvoření univerzálního hostitele může být krajně nepříjemné řešení, protože tříd zařízení je mnoho, existuje řada výjimek a částí, které jsou ponechány na implementaci konkrétního koncového zařízení. V takovém případě se může rozhraní stát nepřehledné, pomalé a snadno může dojít k zanesení chyby při implementaci.

Mé řešení bude zaměřeno na *softwarové rozhraní mezi procesorem a USB hostitelem*. Bude vytvářet obecné aplikační rozhraní, které umožní obsluhu jakéhokoliv USB zařízení na obecné úrovni. Možnosti aplikačního rozhraní budou demonstrovány využitím této mezivrstvy k vytvoření konkrétní aplikace, která bude obsluhovat zařízení třídy CDC (communication devices class). V rámci řešení práce bude vyvíjena snaha o přenositelnost v maximální možné míře, ideálně pouze na úrovni hardware abstrahující vrstvy.

Můj přínos práce do problematiky vidím ve vytvoření přenositelného (na úrovni zdrojového kódu) USB hostitele pro mikrokontroléry, kde se zpravidla liší implementace rodinu od rodiny, výrobce od výrobce. Přenesení na zcela odlišnou platformu by v ideálním případě mělo znamenat pouze vytvoření hardware abstrahující vrstvy pro přístup k příslušným registrům, případně nastavení konfiguračních hlavičkových souborů.

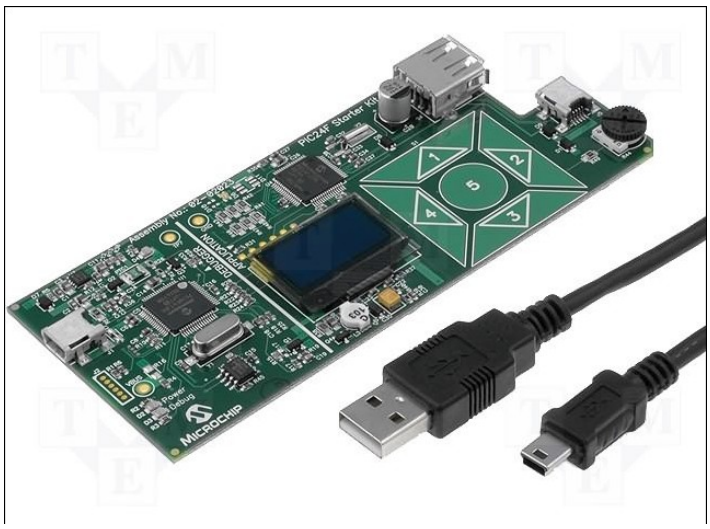
3.1 Seznámení s kitem Microchip pro USB host

3.1.1 MPLAB Starter Kit

Kit Microchip pro USB host (MPLAB Starter Kit for PIC24F MCUs), viz Obrázek 3.1, který mám zapůjčen se skládá z programátoru a samotné aplikace. Aplikační část kitu je založena na mikrokontroléru PIC24FJ256GB106, který je mimo jiné vybaven USB rozhraním. USB rozhraní mikrokontroléru vyhovuje standardu On-The-Go, aplikace tak může sloužit jak koncové zařízení, tak USB hostitel. USB rozhraní může pracovat s přenosovou ryhclostí low-speed i full-speed. Jádrem mikokontroléru je 16 bitový RISC procesor, který při použití USB modulu pracuje na frekvenci 12 MHz. Čip obsahuje 16 kB operační paměti SRAM a 256 kB paměti flash pro data a program. Dle materiálů výrobce je USB modul v režimu hostitele omezen na variantu redukovaného hostitele, tedy je možné připojit pouze jedno koncové zařízení, připojit rozbočovač není možné. Standardně jsou podporována zařízení typu přenosné datové úložiště a existuje i software pro obsluhu zařízení třídy HID (klávesnice, myš). Podporu dalších tříd zařízení je možné doplnit doprogramováním příslušného ovladače, jediným limitujícím faktorem je paměť čipu. Samozřejmě je možné dodaný firmware zcela nahradit vlastní aplikací a rozšířit tak možnosti výsledné aplikace.

Kit dále obsahuje OLED displej (128x64 bodů, monochromatický), LED diody řízené pomocí PWM výstupů, klávesnici s pěti kapacitními dotykovými klávesami a potenciometr. Od výrobce je dodávána široká škála programů, demo aplikací a různých knihoven, včetně řady knihoven pro USB. Knihovny pro USB mimo jiné obsahují holý USB host stack. Pro práci s displejem je přítomna grafická knihovna. Přítomnost knihoven pro obsluhu dalších periferií je důležitá pro rychlý, pohodlný a kvalitní vývoj koncové aplikace, kdy je možné soustředit veškeré snažení na cílovou aplikaci a není třeba vyvíjet a ladit podpůrné části kódu.

Dodaný USB host stack provádí proces enumerace a poskytuje aplikační rozhraní pro čtení a zápis dat na úrovni koncových bodů. Aplikace však slouží především pro účely demonstrace možností čipu a nevhodným způsobem řeší některé chybové stavy a výběr koncových bodů pro provedení komunikace. Koncové body připojeného zařízení jsou řazeny do lineárního seznamu, který je vždy procházen od začátku, pokud tedy nějaký koncový bod ze začátku jednosměrně vázaného seznamu opakovaně vyvolává chybu, kterou je třeba ošetřit, tak se na obsluhu koncových bodů na konci seznamu nemusí dostat. Zdrojový kód poměrně těžko čitelný, navíc podpora některých druhů zařízení je umožněna jen díky grafické aplikaci (myslím, že tato grafická aplikace je dostupná pouze pro operační systém MS Windows), která generuje objektový kód, což je sice vhodné pro rychlé prototypování nebo ozkoušení funkcí bez znalosti detailů ohledně provozu na USB sběrnici, není to však vhodné z hlediska kontroly nad výsledným kódem, což bývá při nasazení v praxi poměrně velký problém. Největší nevýhodou dodaného kódu je velmi úzká vazba na cílový hardware, tudíž je pro plánovanou aplikaci v podstatě nepoužitelná a při vývoji bude třeba začínat od nuly.



Obrázek 3.1: Vývojový kit MPLAB Starter Kit pro PIC24F.

Obrázek převzat z [15].

3.1.2 Vývojové prostředí MPLAB

Vývojové prostředí MPLAB je typickým zástupcem integrovaného vývojového prostředí (IDE – integrated development environment), které zastává řadu funkcí, například správce souborů zdrojových kódů projektu, textový editor, překlad a sestavení programu, organizace paměti výsledné aplikace, ladicí nástroje a tak dále, některé funkce budou dále popsány podrobněji.

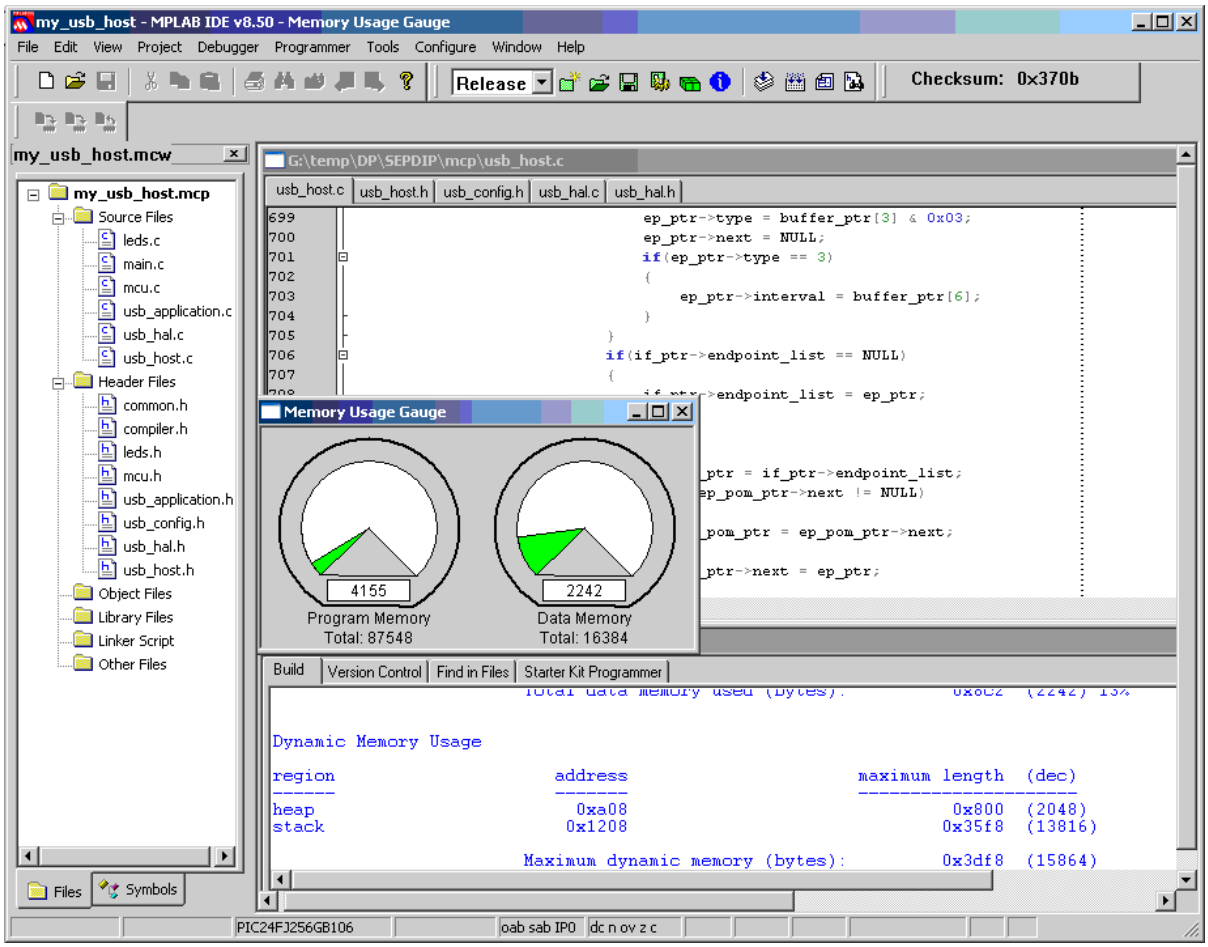
Textový editor automaticky zvýrazňuje zdrojový kód a umožňuje sbalení, respektive rozbalení bloků kódu, čímž výrazně přispívá k snadné orientaci v rozsáhlejších souborech zdrojových kódů. Drobným nedostatkem je, že si prostředí neuchovává předchozí stav sbalení/rozbalení textu, takže je třeba při každém znovuotevření souboru provádět požadované sbalení.

Pro překlad a sestavení programu jsou volány externí programy, které jsou však s prostředím standardně dodávány. Prostředí obsahuje podpůrné nástroje pro rychlé nastavení konfiguračních bitů mikroprocesoru, pro některé rodiny čipů umožňuje i nastavení periferií pomocí grafického rozhraní. Neplatí to však pro rodinu PIC24FJxx s USB rozhraním. Dále prostředí obsahuje například nástroj Memory Usage Gauge viz Obrázek 3.2, který poskytuje intuitivním způsobem přehled o využití paměti mikrokontroléru, nebo pohled na obsah programové paměti a mnoho dalších funkcí.

Vývojové prostředí rovněž umožňuje ladění vyvíjené aplikace, včetně obvyklých funkcí krokování po instrukci skok do bloku, přeskočení bloku, reset procesoru a jiné. Bohužel však nelze těchto nástrojů využít pro ladění funkcí pracujících s USB modulem. Ke správné funkci USB modulu je totiž zapotřebí plné rychlosti procesoru, v režimu ladění jsou přerušení generovaná USB modulem maskována.

Velkou výhodou vývojového prostředí je jeden program, jedno shodné prostředí pro všechny produkty Microchip, liší se pouze použitý překladač a sestavovací program. Vývojové prostředí navíc zvládne obsluhu připojených programátorů, takže k nahrání firmwaru do mikroprocesoru není třeba dalších aplikací (např. integrované vývojové prostředí AVR Studio pro 8 bitové čipy toto neumožňuje a pro samotné naprogramování čipu je potřeba použít jiný program). Nevýhodou vývojového prostředí jsou problémy s vývojovými kity, pokud dojde ke spuštění více než jedné instance programu. Vývojový kit je přehrán novým firmwarem a uvedení kitu do původního stavu je možné

pouze ručním propojením pinů na desce plošných spojů vývojového kitu. Věřím však, že jde o dočasný nedostatek, který bude v některé z pozdějších verzí odstraněn.



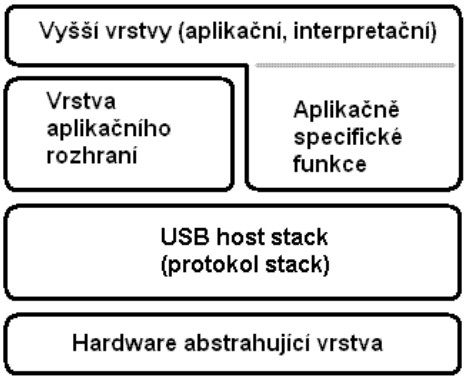
Obrázek 3.2: Vývojové prostředí MPLAB (vlevo správce projektů a zdrojových kódů projektu, dole výpis konzole při překladu, v horní části textový editor, v popředí monitor využití paměti).

4 Návrh systému

Obecné požadavky na návrh systému vycházejí z definice úkolu uvedené v kapitole 3. Konkrétně se jedná o obecné softwarové rozhraní USB hostitele, přenositelné na úrovni zdrojových kódů. Aplikace bude rozčleněna do více vrstev, možné rozdělení vrstev viz Obrázek 4.1. Rozdělením do vrstev se zjednoduší případná modifikace nebo rozšíření funkčnosti dle potřeb a zajistí se tím přenositelnost s výjimkou hardware abstrahující vrstvy, která je už z podstaty obecně nepřenositelná. Na základě zmíněného rozdělení vrstev bude vytvořena aplikace, která bude komunikovat se sériovým komunikačním zařízením typu modem, telefon.

Mezi podporované typy přenosů patří nezbytný řídicí přenos (control), vyzývaný přenos (interrupt) a běžný přenos (bulk), podpora isochronního přenosu není v návrhu zahrnuta. Návrh systému je rozdělen do několika oddělených vrstev. Vyšší vrstvy mohou obecně přistupovat k více nižším vrstvám zároveň, protože nelze jednoduše zakázat volání funkcí nižší vrstvy, návrh aplikace však taková volání neobsahuje. Funkce konkrétní vrstvy jsou volány pouze uvnitř této vrstvy nebo z nejbližší vyšší vrstvy.

Aplikace je navržena pro snadné použití jak v systémech bez operačního systému, tak pro systémy s řízeným přepínáním úloh, například pomocí operačního systému pro vestavěné systémy. Z tohoto důvodu je třeba zajistit, aby vrstva USB host stacku mohla běžet navenek paralelně s hlavní aplikací. Tím bude zajištěno lepší využití výpočetního výkonu s důrazem na rychlost odezvy. Pro rychlou odezvu a korektní běh je třeba rozčlenit provádění vrstev do menších funkčních celků, vyvarovat se aktivního čekání a zajistit mechanismus pro předávání informací a příkazů mezi jednotlivými vrstvami. Paralelního běhu bude dosaženo prostým střídáním úloh rozdělených na kratší úseky, alternativní možnosti řešení jsou podrobněji popsány v kapitole 5.1, kde jsou diskutovány výhody a nevýhody alternativních řešení včetně řešení prioritizace jednotlivých vrstev.



Obrázek 4.1: Rozdělení vrstev systému.

4.1 Hardware abstrahující vrstva

Tato vrstva má za úkol oddělit specifické vlastnosti hardwaru od obecného softwarového rozhraní, slouží pro přístup k registrům mikrokontroléru pro obsluhu USB rozhraní a k obsluze přerušení USB modulu. Tato vrstva bude jako jediná závislá na použité platformě. Všechny vyšší vrstvy systému přistupují k hardwaru výhradně pomocí této vrstvy. Pro případ softwarové emulace fyzického USB

rozhraní nebo omezených možností hardwaru poskytuje tato vrstva vyrovnávací paměť koncových bodů. Úplná absence vyrovnávací paměti v hardwaru s podporou USB je málo pravděpodobná, ale vyloučit ji nelze. U některých rodin mikrokontrolérů je vyrovnávací paměť pro USB modul dedikovaná – tzn. tuto paměť nelze použít pro jiné účely, toto je třeba příklad 8 bitových mikrokontrolérů Atmel AVR. Další možností vyrovnávání paměti je vyhrazení určitého rozsahu běžné operační paměti pro USB modul. Zcela jiný způsob přístupu do paměti USB modulem poskytuje řešení použité u mikrokontrolérů PIC 24F, kde je přístup do paměti řešen pomocí řadiče přímého přístupu do paměti. Přítomnost vyrovnávací paměti hardware abstrahující vrstvy výrazně ulehčí obsluhu z nadřazených vrstev a sníží zatížení procesoru. U mikrokontrolérů rodiny PIC24F toto není třeba řešit díky již zmíněnému řadiči přímého přístupu do paměti, který bude detailněji popsán v následujícím odstavci.

Mikrokontroléry PIC využívají zajímavou techniku pro komunikaci mezi USB modulem a operační pamětí. Základním prvkem je tabulka struktur popisujících vlastnosti koncových bodů (tzv. buffer deskriptory) USB modulu v mikrokontroléru. Tuto tabulku je možné vytvořit kdekoliv v operační paměti, jen je třeba začátek této tabulky zarovnat na začátek stránky v paměti, jinými slovy je třeba, aby adresa začátku tabulky byla celým násobkem 512. Při inicializaci USB modulu se nastaví tato adresa do příslušného registru. Samotné položky tabulky slouží k nastavení parametrů přenosu před jeho zahájením, po dokončení přenosu je položka aktualizována stavem dokončené operace. Buffer deskriptory obsahují dvě hlavní položky – adresu a kopii registru kontroléru. Kopie registru slouží k uchování informace po dokončení přenosu, je tak možné zahájit další přenos na jiném koncovém bodu bez předchozího zpracování stavu dokončené operace. Adresa slouží k uložení adresy bufferu pro přenos dat během přenosu. Toto je klíčová vlastnost, která umožňuje u mikrokontrlérů této rodiny u hardware abstrahující vrstvy vynechat alokaci vyrovnávací paměti, protože díky této adrese může s využitím řadiče přímého přístupu do paměti USB modul přistupovat k jakékoliv adrese operační paměti, libovolně zarovnané a na tuto adresu zapisovat nebo z ní číst.

Pokud je USB hostitel schopen komunikace na různých rychlostech, poskytuje tato vrstva informaci o rychlosti využívané připojeným zařízením. Komunikační rychlost připojeného zařízení je generována nastavením určité napěťové úrovně na datových vodičích. Rovněž tato vrstva generuje časování a značky začátků rámců (Start of Frame – SOF), případně další služby pokud je jich třeba, zpravidla však toto bývá implementováno přímo v hardwaru mikrokontroléru a hardware abstrahující vrstva slouží pouze k obsluze této funkcionality.

4.2 USB host stack

Vrstva USB host stack obsahuje funkce pro zpracování příznaků generovaných v hardware abstrahující vrstvě a poskytuje funkce pro komunikaci na úrovni koncových bodů. Umožňuje zasílání dat z hostitele do zařízení a získávání dat ze zařízení. V rámci této vrstvy je implementován proces enumerace. Proces enumerace se omezuje na standardní funkce, aplikačně specifická rozšíření by byla umístěna umístěna v dalším modulu, pokud by byla požadována, viz Obrázek 4.1. Proces enumerace je podrobně popsán v podkapitole 4.2.1.

Přístup ke koncovým bodům umožňuje jednak zasílání dat do komunikačních koncových bodů, jednak výměnu informací pomocí řídicího koncového bodu, protože některá zařízení využívají řídicí koncový bod k nastavení vnitřních parametrů nezbytných pro správnou funkci, případně některá

jednoduchá zařízení využívají pouze řídicího koncového bodu – koncový bod 0, který je přítomen v každém USB zařízení (zařízení, která využívají pouze napájení z rozhraní, jako například LED lampičky se neberou v potaz).

V rámci implementovaného systému se bude pro přístup k hardware abstrahující vrstvě využívat pouze tato vrstva, přestože obecně je možné přistupovat k nižším vrstvám přímo, přímý přístup k více vrstvám představuje překázku přenositelnosti, proto této vlastnosti nebude využito. Nad vrstvou USB hostitele je možné vystavět ještě další vrstvy, které budou automatizovat některé dílčí záležitosti, například tak bude možné vytvořit rozhraní pro proudové zasílání dat například ze sériového portu bez nutnosti dělení proudu dat do zpráv manuálně nebo vytvořit zapouzdření pro specifické aplikace ve stylu ovladače zařízení, jako je tomu u osobních počítačů. Nadstavbovou vrstvou bude vrstva aplikačního rozhraní, která kromě zapouzdření funkcí pro obsluhu komunikace koncových bodů bude obsahovat funkce pro pohodlné vyhledání zařízení mezi připojenými zařízeními. Obsluha z vyšších úrovní skrývá před programátorem hlavní aplikace systém organizace připojených zařízení k USB hostiteli, způsob kontroly dokončení operací a další služby, které zpravidla výrazně prodlužují vývoj aplikace.

Jak bylo zmíněno v úvodu této kapitoly, je třeba zajistit pseudoparalelní běh vrstvy USB host stacku a aplikačních vrstev. Návrh aplikace počítá s rozdělením hlavní funkce USB host stacku na kratší úseky. Mezi jednotlivými voláními kratších úseků dojde k prostřídání s aplikační úlohou. Ideálním řešením tak bude vytvoření stavového stroje s pamětí pro stav a další přidružené struktury, ve kterých budou uchovány nezbytné informace mezi jednotlivými voláními hlavní funkce této vrstvy. Kromě střídání běhu úloh je třeba zajistit předávání informací s ostatními vrstvami. Pro komunikaci USB host stack ↔ API (vrstva aplikačního rozhraní) bude k této výměně informací sloužit fronta požadavků, kterou bude USB host stack zpracovávat po úspěšné úvodní konfiguraci připojeného zařízení. Průběžné zpracování fronty požadavků spočívá ve zpracování tohoto požadavku, následované voláním hadrware abstrahující vrstvy k provedení operace přenosu. Poté přejde stavový stroj do stavu čekání. Čekání je ukončeno se změnou sdíleného příznaku hardware abstrahující vrstvy, viz dále, které vyvolá zpracování dokončeného přenosu – zpracování stavu operace a aktualizace položky ve frontě zpráv, včetně posunu čela frotny na další položku.

S hardware abstrahující vrstvou probíhá výměna informací pomocí sdíleného příznaku o zpracovaných přerušeních hardware abstrahující vrstvou, vrstva USB host stacku podle přijatého příznaku provede příslušné akce, například vyčtení stavu dokončené operace pomocí funkcí hardware abstrahující vrstvy.

4.2.1 Proces enumerace

Pod pojmem enumerace se rozumí sled akcí potřebný k detekci a konfiguraci připojeného zařízení USB hostitelem. Jednotlivé akce vycházejí ze stavového diagramu zařízení, uvedeného v kapitole 9 dokumentu USB 2.0[16], str. 240.

Po připojení zařízení ke sběrnici nebo po spuštění napájení sběrnice je zařízení v *attached* stavu. V tomto stavu zařízení nereaguje na jakýkoliv provoz na sběrnici, pouze je napájeno (maximální odběr proudu je limitován standardem). USB hostitel poté čeká minimálně 100 ms na dokončení zasunutí zařízení uživatelem k ustálení napájení připojeného zařízení, zařízení přejde do stavu *powered*. Pokud se tak nestalo již dříve, hostitel spustí generování začátků rámců, tím je

zajištěn provoz na sběrnici, který je pro zařízení rozhodující pro ne/přepnutí do režimu snížené spotřeby.

Hostitel poté zjistí komunikační rychlost připojeného zařízení sejmutím napěťových úrovní na datových vodičích. Zařízení typu low-speed zvyšuje napětí pomocí pull-up rezistoru na vodiči D-, zařízení typu full-speed zvyšuje napětí na vodiči D+. Po zjištění rychlosti zařízení hostitel provede reset zařízení uvedením sběrnice do stavu resetu (oba vodiče do stavu log. 0) po dobu minimálně 10 ms, po uvolnění sběrnice ze stavu resetu se zařízení nachází ve výchozím (*default)* stavu.

Ve výchozím stavu má zařízení výchozí adresu 0 a naslouchá příkazy na řídicím koncovém bodě 0. Pomocí koncového bodu 0 přijímá příkazy a příslušným způsobem na ně reaguje. Zařízení musí implementovat minimálně příkazy pro získání deskriptorů a nastavení adresy. Typický průběh při enumeraci ve výchozím stavu obsahuje zaslání požadavku pro zaslání prvních 8 bajtů deskriptoru zařízení viz podkapitola 2.3.4. V prvních 8 bajtech je jednak informace o velikosti řídicího koncového bodu, jednak velikost celého deskriptoru zařízení. Následně je proveden příkaz pro zaslání celého deskriptoru zařízení. Ve zbývající části deskriptoru zařízení jsou ID výrobce a produktu zařízení, včetně případného sériového čísla. Tyto informace mohou v některých případech ovlivnit průběh zbytku procesu enumerace, případně jej zcela ukončit, podrobněji o tomto problému na konci podkapitoly po dokončení popisu procesu. Z výchozího stavu je zařízení převedeno do adresovaného (*address*) stavu převedeno pomocí příkazu *set_address*, kterým je zařízení přiřazena adresa jednoznačná na sběrnici. Hostitel po potvrzení příkazu zařízením čeká 2 ms ke změně adresy v zařízení.

Adresovaný stav slouží pro vyčtení všech ostatních deskriptorů potřebných k dokončení konfigurace zařízení. Proces se opět běžně skládá z více fází, většinou slouží k vyčtení deskriptoru konfigurace. Ten někdy bývá opět rozdělen na více částí, typicky vyčtení prvních 9 B deskriptoru konfigurace, kde 8. a 9. bit udává jeho celkovou délku. Poté je zaslán příkaz pro vyčtení celého deskriptoru. Jakmile je získán celý deskriptor, má hostitel veškeré potřebné informace ke konfiguraci svého vnitřního stavu, případně mohou být ještě požadovány textové deskriptory, které již nejsou pro konfiguraci nezbytné. Zasláním příkazu *set_configuration*, je zařízení uvedeno do stavu *configured*, kdy je plně funkční a odpovídá na požadavky na všech koncových bodech (ve všech předchozích stavech odpovídá pouze na požadavky pro koncový bod 0).

Proces enumerace v operačních systémech MS Windows má trochu odlišný průběh, který může při vývoji koncových zařízení činit začínajícím vývojářům potíže. Ve výchozím stavu je zaslán zařízení požadavek na zaslání 255 B deskriptoru zařízení, avšak po přenesení pouhých 8 B hostitel resetuje sběrnici, po resetu sběrnice znovu žádá o zaslání deskriptoru zaříení, tentokrát již však požaduje správnou délku 18 B, což je délka tohoto deskriptoru. Poté již probíhá proces enumerace výše popsaným způsobem.

Informace deskriptoru zařízení o výrobci a produktu (VID, PID, sériové číslo) mohou vést k odmítnutí dalších akcí s tímto zařízením ze strany hostitele, protože už nyní může být jasné, že vyšší vrstvy nedovedou/nechtějí takové zařízení obsloužit. V produkční oblasti je tohoto často využíváno k zabezpečení neautorizovaného užití zařízení s vestavěným USB hostitelem (například upgrade firmware je možný pouze s využitím konkrétního koncového zařízení) nebo k donucení zákazníka ke koupi uceleného specializovaného systému, přestože oním specializovaným zařízením je obyčejný flash disk s maskovanými deskriptory. S tímto jsem se osobně setkal u monitorovacího zařízení v automobilech, ze kterého je potřeba pravidelně získávat záznamy a tyto posílat na ústředí. V tomto případě byl použit obyčejný flash disk s maskovanými deskriptory – v počítači je tak obsluha

takového zařízení možná pouze s využitím dodaného programu a zařízení v automobilu na jiné zařízení záznam nenahraje. Zákazník tak platí nejen za samotné zařízení v automobilu, ale i za flash disk (řádně oceněný) a obslužný program.

Proces enumerace není přesně definován, proto se může lišit podle operačního systému nebo podle specifického použití, jak již bylo zmíněno výše. Extrémním příkladem je přizpůsobení aplikace jednomu konkrétnímu zařízení, v případech, kdy je zajištěno, že bude připojeno vždy toto a pouze toto zařízení. Tato situace nastane třeba při použití zařízení i hostitele na jedné desce plošných spojů, například připojení GSM modulu v ústředně odlehlého zařízení, například dálkově ovládaného zabezpečení objektu. V takovém případě je možné omezit proces enumerace na sled restart sběrnice, přidělení adresy a volba konfigurace. Protože jsou předem známy parametry takového zařízení, je možné všechny potřebné informace – třeba čísla koncových bodů, umístit přímo do aplikace a není třeba získávat deskriptory.

4.3 Vrstva aplikačního rozhraní

Vrstva aplikačního rozhraní zapouzdřuje některé funkce USB hostitele a zjednodušuje obsluhu ze strany aplikace. Zejména se to týká funkcí pro čtení a zápis dat, kde aplikace provede jedno volání funkce aplikační vrstvy a tato funkce se postará o další zpracování dle potřeb, protože pro jednu transakci přenosu dat je obecně třeba více volání, je třeba přenést několik paketů k dokončení transakce. Například nastavení parametru zařízení pomocí řídicího přenosu se skládá ze třech fází – úvodní fáze, která nastavuje parametry přenosu, druhá fáze přenáší data a poslední fáze potvrzuje správné přijetí dat druhou stranou. Nad vrstvou aplikačního rozhraní je možné vytvořit další vrstvy v rámci výsledné obslužné aplikace. Tyto vrstvy mohou mít například podobu ovladačů, které zpracovávají data z/do formátu koncového zařízení.

Všechny požadavky na přenos dat jsou vrstvě USB host stacku předávány pomocí fronty zpráv. Vkládání a vyjímání položek z fronty se děje pomocí funkce vrstvy USB host stacku. Každá položka z této fronty obsahuje jednoznačnou identifikaci zařízení a koncového bodu, se kterým má komunikace probíhat, typ požadovaného přenosu a stav prováděné operace. Ve frontě může být v jeden okamžik pouze jeden požadavek na konkrétní koncový bod. Je to z toho důvodu, že je třeba testovat úspěšnost dokončení předchozí operace, případně provést zotavení z chyby. USB host stack po dokončení konfigurace zařízení po jeho připojení tuto frontu požadavků začne zpracovávat. Po dokončení přenosu je položka fronty zpráv vrstvou USB host stacku aktualizována, aby vrstva aplikačního rozhraní mohla příslušným způsobem reagovat. Reakcí je zde míněno převzetí stavu předchozí operace, vyjmutí dokončené operace z fronty a podle potřeby je vložen další požadavek, pokud nebyla přenesena všechna data nebo do dokončení přenosu vznikla nová data k přenesení.

Kromě funkcí pro přenos dat obsahuje vrstva aplikačního rozhraní funkce pro zpracování informací o stavu sběrnice, o připojených zařízeních, rozhraních a koncových bodech v rámci zvolené konfigurace připojených zařízení, obecně funkce pro zpracování datových struktur vrstvy USB host stacku. Tyto funkce výrazně zjednodušují hlavní aplikaci, zvyšují tak její přehlednost a snižují riziko programátorských chyb. Z navrhovaných funkcí bych zmínil například funkci, která vyhledá připojené zařízení podle jeho třídy, nadřazená aplikace tak pomocí jednoho volání získá jednoznačnou adresu připojeného zařízení, pokud je zařízení vyhledávané třídy v systému přítomné. Aplikace tak nemusí řešit, zda je k hostiteli nějaké zařízení připojeno, pokud je připojeno, jestli je

konkrétní třídy, pokud je konkrétní třídy, jestli je nakonfigurováno a připraveno k použití. Obdobná funkce je i pro vyhledání indexu rozhraní podle třídy v rámci konfigurace. Index rozhraní poté slouží k získání čísla koncového bodu, který je klíčovým prvkem při adresování komunikace na sběrnici.

Mezi další nezbytné funkce patří získání směru komunikace konkrétního koncového bodu, nyní popíšu využití této funkce. Aplikace vyhledá zařízení třídy CDC, pokud je zařízení nalezeno, vyhledá rozhraní třídy CDC_COMM, získá číslo koncového bodu, protože je v tomto rozhraní standardně pouze jeden koncový bod, je jeho směr předem známý, avšak v následně vyhledaném rozhraní třídy CDC_DATA jsou koncové body dva, jeden pro komunikaci směrem hostitel → zařízení, druhý pro směr zařízení → hostitel. Pomocí funkce pro získání směru koncového bodu se určí, který je který.

4.4 Vrstvy nadřazené aplikačnímu rozhraní

Jak bylo uvedeno v podkapitole 4.3, je možné nad vrstvou aplikačního rozhraní vytvořit další vrstvy. V rámci implementované aplikace bude pro demonstraci možností vytvořeného systému přítomna vrstva aplikace pro komunikaci pomocí jednoduché sady příkazů, která bude umožňovat komunikovat s připojeným sériovým zařízením, zasílat tomuto zařízení příkazy a data a získávat z tohoto zařízení jeho stav a data. Kromě vrstvy samotné aplikace bude nutné vytvořit mezivrstvu pro interpretaci příkazového jazyka.

Tyto vyšší vrstvy obecně musí průběžně kontrolovat, zda nedošlo k odpojení zařízení, se kterým komunikují, aby aplikace neuvázla v nějakém stavu bez možnosti znovuuvedení aplikace do definovaného stavu. Toto je nezbytné pro správnou funkci při opětovném připojení, lze totiž předpokládat, že běh takového systému může obecně trvat velmi dlouhou dobu bez restartu. Mnohé vestavěné aplikace běží mnoho měsíců bez jediné odstávky. Pokud by to časové důvody nedovolily, je možné vynechat interpretační vrstvu a založit demonstrační aplikaci přímo nad vrstvou aplikačního rozhraní, k předvedení možností USB hostitele a aplikačního rozhraní by to bylo dostatečné.

4.4.1 Interpretační vrstvy

Interpretační vrstvy budou zpracovávat příkazový jazyk aplikace, který bude mezivrstva interpretovat na volání funkcí aplikačního rozhraní. Příkazových jazyků může být více. Jeden může obsahovat příkazy ve tvaru ASCII, aby bylo možné snadno komunikovat se systémem pomocí terminálu srozumitelnými příkazy, druhý jazyk může obsahovat příkazy v hexadecimálním tvaru pro automatizovanou komunikaci pomocí programu. Hexadecimální podoba příkazů je výhodná jednak pro snížení objemu nezbytné komunikace, jednak pro odstranění vícenásobného inverzního převodu například při předávání číselných informací, kdy je možné ušetřit až dvě třetiny objemu přenášených dat a snížit výpočetní zátěž díky možnosti přímé manipulace s přijatými daty.

4.4.2 Aplikační vrstva

Aplikační vrstva bude v navrhovaném systému nejvyšší vrstvou, kromě zasílání holých dat bude umožňovat zasílat AT příkazy do zařízení. Pro případ připojení mobilního telefonu k USB hostiteli bude aplikační vrstva umožňovat poslat SMS zprávu nebo vytočit telefonní číslo. Vrstva bude využívat výhradně interpretační vrstvy pomocí sady příkazů.

Tuto část aplikace bude opět vhodné řešit jako stavový stroj pro rozmělnění obsluhy do kratších úseků. Před samotnou komunikací s připojeným zařízením je třeba nejprve ověřit, zda je připojeno některé z podporovaných zařízení. Po detekci připojení zařízení a jeho výchozí konfiguraci vrstvou USB host stacku aplikace s využitím aplikační vrstvoy získá adresu tohoto zařízení, pomocí adresy pak parametry rozhraní a koncových bodů pro další komunikaci, například se ověří, zda zařízení implementuje potřebnou sadu příkazů k nastavení parametrů linky a podobně. Po ověření a nastavení potřebných parametrů bude vhodným způsobem demonstrována funkční komunikace mezi aplikací hostitele a koncovým zařízením.

Aplikační vrstva musí vhodným způsobem detekovat případné odpojení zařízení. Při využití interpretační vrstvy se nabízí zaslání informace o změně stavu po dokončení probíhajícího příkazu interpretační vrstvou. Pokud by interpretační vrstva nebyla využita, musí aplikační vrstva aktivně testovat změnu stavu generovanou USB host stackem pomocí funkcí aplikačního rozhraní.

5 Implementace systému

Implementace bude probíhat kompletně v jazyce C s využitím vývojového prostředí Microchip MPLAB IDE a nástrojů překladače MPLAB C30 C Compiler. Bude následovat popis jednotlivých vrstev, bude popsáno, jak je co konkrétně realizováno, budou konzultovány možné přístupy k řešení problému a zdůvodněno vybrané řešení. Dále je popsáno, jaké HW a SW prostředky jsou využity, případně vývojové diagramy.

Zdrojové kódy budou využívat pojmenování identifikátorů a komentářů v anglickém jazyce. Použití anglického jazyka ve zdrojových kódech není bezúčelné, předpokládá se totiž, že anglicky bude rozumět každý, kdo se o danou problematiku zajímá. Tím je umožněno porozumění koncepci a konkrétnímu řešení například zahraničním studentům, kteří by s česky komentovaným kódem mohli mít potíže, rozšiřuje se tak obor případných čtenářů a zájemců o nastudování tohoto řešení.

5.1 Koncepce

Koncepce celého systému vychází z několika předpokladů. Nejdůležitějším předpokladem je důraz na přenositelnost, s tím souvisí nutnost oddělit hardware s obsluhou přerušení od jádra aplikace, kterým je v případě implementovaného systému vrstva USB host stacku. Důležitá je nezávislost vrstvy aplikace a vrstvy USB host stacku. Je sice možné řešit aplikaci tak, že po nastavení každého požadavku program čeká na jeho dokončení před provedením dalších operací, je to však krajně nevhodné řešení, které v podstatě znemožňuje rozšiřitelnost takto vytvořené aplikace. Navíc to absolutně nezapadá do konceptu, kdy je snahou vytvořit podpůrnou vrstvu pro univerzální použití. Ve chvíli, kdy se čeká na dokončení nějaké operace, například přenos paketu na USB rozhraní, je možné provést spoustu užitečné práce na jiných komunikačních rozhraních nebo tento čas využít ke kontrole mnohých vstupů a zpracování již přijatých dat, jak je to v oblasti vestavěných systémů běžné.

Dalším předpokladem je možnost nasazení USB hostitele jak v systému bez operačního systému a bez řízení přepínání úloh plánovačem (tak to je řešeno v implementovaném systému), tak v aplikaci s operačním systémem. Aplikace s operačním systémem pro vestavěná zařízení mají specifický způsob ošetření přerušení – samotné rutiny přerušení jsou minimalistické, obsahují pouze nezbytně nutný kód a signalizují příchod přerušení pomocí sdíleného příznaku, který je pravidelně kontrolován nadřazenou úlohou a následně touto úlohou dochází ke zpracování podle nastavených příznaků, případně přijatých dat.

Způsob minimální obsluhy v rutině přerušení může. ale nemusí, znamenat o něco nižší výkonnost oproti řešení založenému výhradně na obsluze v rutině přerušení (to je třeba příklad USB host stacku dodávaného mezi vzorovými příklady k vývojovému kitu). Mezi faktory snižující výkonnost patří prodlevy mezi vznikem požadavků a jejich skutečným obsloužením. Prodlevy jsou dány dvěma faktory, prvním je potřeba předání řízení funkci, která tyto příznaky zpracovává, druhým je samotné nastavení těchto příznaků. Je třeba tyto příznaky nastavit, v hw registrech kontroléru nulovat příznaky přerušení a v obsluhující funkci je následně po obsloužení třeba nulovat sdílený příznak, který je často určen ukazatelem nebo nepřímým přístupem do paměti. Na druhou stranu přináší takové řešení lepší čitelnost kódu díky striktnímu oddělení od hardwaru a hlavně výrazně zkracuje dobu strávenou v obsluze přerušení, nedochází tedy ke stárnutí obsluhy přerušení na nižších

prioritních hladinách. Minimální obsluha v rámci přerušení má rovněž výhodu v tom, že ke složitější obsluze s testováním řady různých příznaků dochází ve vyšších vrstvách pouze v případě požadavků na přenost dat, které jsou spouštěny explicitně. Samotná obsluha bez datového provozu na sběrnici se tak omezuje pouze na několik jednoduchých podmínek, které vytěžují procesor maximálně po několik desítek instrukčních cyklů.

V implementovaném systému není explicitně řešeno přepínání úloh, úloha USB hostitele a aplikace nad ním vystavěná se pravidelně střídají v nekonečné smyčce, jedna po dokončení druhé, viz Kód 5.1. Obě úlohy jsou řešeny jako stavové stroje, čímž umožňují rozmělnění obsluhy do krátkých úseků, ke střídání úloh (tím i obsluze USB modulu) tak dochází dostatečně často. Výhodou tohoto řešení je jednoduchost řešení. Pokud by však nadřazená aplikace byla složitější nebo neumožňovala pravidelné přepínání, například z důvodu aktivního čekání při obsluze jiných částí nebo periferií, je možné toto jednoduše ošetřit. Zmíněné ošetření by spočívalo ve spuštění časovače s vhodně nastavenou periodou přerušení, časovač by pracoval s prioritou přerušení nižší než je přerušení od USB modulu. Rutina obsluhy přerušení takto nastaveného časovače by spouštěla úlohu USB hostitele. Z hlediska nadřazené aplikace by se tak USB hostitel jevil jako úloha spuštěná paralelně v pozadí. Vhodně zvolená perioda takového časovače je klíčová, pokud by tato byla příliš krátká, docházelo by pouze k obsluze USB hostitele, naopak příliš dlouhá perioda vede na plýtvání časem a tím omezení přenosové šířky pásma.

Pokud by nebyl možný výše popsaný jednoduchý způsob volání pomocí časovače, například pro nedostupnost volného časovače, je stále možné manipulovat s rozdělením priorit běhu úloh jednotlivých vrstev. V takovém případě je však nezbytné rozdělení aplikační úlohy do menších částí, například pomocí koncepce stavového stroje. Řešení spočívá v opakovaném volání zvýhodňované úlohy v hlavní programové smyčce celé aplikace, viz Kód 5.1. Výhodou tohoto řešení je zvýšení priority některé úlohy bez implementace přepínání úloh uprostřed jejich běhu.

Pro přehlednost provedu shrnutí možných řešení bez explicitního přepínání úloh. Prosté střídání 1:1 – tak je to v demonstrační aplikaci. Střídání M:N – princip demonstrační aplikace se změnou priority zvýhodňující některou z aplikací (pravděpodobně by byl zvýhodněn USB host stack). Běh hlavní aplikace v popředí a USB host stack volán pomocí přerušení časovače s vhodně nastavenou prioritou. Při použití operačního systému s přepínáním úloh je vždy třeba přihlédnout na princip mechanismu přepínání úloh takového systému a podle toho upravit aplikaci, například zabránění přepnutí úloh v některých pasážích kódu.

```
usb_host_init();
usb_application_init();

for(;;)
{
      usb_host_task();
      usb_application_task();
}
```

Kód 5.1: Jádro hlavní aplikace - volání úloh.

5.2 Hardware abstrahující vrstva

Hardware abstrahující vrstva je pojata jako pouhé odstínění hardwaru bez výrazné vlastní funkcionality, většina procesní logiky je až ve vyšších vrstvách. K dosažení větší znovupoužitelnosti veškerého kódu mají všechny funkce hardware abstrahující vrstvy jako jeden z parametrů číslo portu. Číslo portu umožňuje obsluhu více samostatných USB portů, pokud by jimi byl mikrokontrolér vybaven. Mikrokontrolér PIC24F má pouze jeden USB port, proto k reálnému využití tohoto parametru v implementovaném systému nedochází, je pouze testován, aby se vyloučilo irelevantní volání.

Mezi klíčové vlastnosti této vrstvy patří inicializace USB modulu mikrokontroléru po jeho spuštění. Mikrokontroléry rodiny PIC24F využívají pro přenosy mezi USB modulem a operační pamětí řadiče přímého přístupu do paměti, tudíž není třeba alokovat vyrovnávací buffery v hardware abstrahující vrstvě i ve vyšších vrstvách, ale stačí před každým přenosem nastavit ukazatel do paměti, toto nastavení neprovádí uživatel před provedením operace čtení nebo zápisu, ale provede se skrytě ve vrstvách podřízených aplikaci. Při inicializaci je třeba nastavit adresu tabulky buffer deskriptorů, viz podkapitola 4.1. Tato tabulka je definována jako globálně definované pole pevné velikosti, jehož adresa je pomocí direktivy překladače zarovnána na začátek stránky v operační paměti, viz Kód 5.2.

```
/* array of buffer descriptors */
BDT_ENTRY __attribute__ ((aligned(512))) BDT[2];
```

Kód 5.2: Definice tabulky buffer deskriptorů.

Kromě inicializace vrstva obstarává nastavení registrů před jednotlivými přenosy, přičemž se nastavení liší podle druhu požadovaného přenosu, a zpracovává přerušení USB modulu. Přerušení jsou vyšším vrstvám signalizována pomocí sdíleného příznaku, nejedná se však o příznak v hodnotě registru mikrokontroléru, protože registry jsou pro každou platformu řešeny různým způsobem, často rozdílně mezi jednotlivými rodinami téhož výrobce. Samotná rutina pro obsluhu přerušení je společná pro všechny druhy přerušení generované USB modulem a neobsahuje veliké množství prováděných operací. Rutina slouží ve většině případů pouze pro obslužení nezbytně nutných operací jako převzetí stavu dokončené operace a podobně, toto chování je v souladu s koncepcí popsanou a zdůvodněnou v podkapitole 5.1. Zajímavostí je nulování příznaků přerušení zápisem logické 1 na příslušnou bitovou pozici v registru. Není tedy možné použít typu instrukcí čtení-modifikace-zápis (read-modify-write), ale je třeba zapsat logickou 1 pouze na té bitové pozici, která přerušení vyvolala, aby nedošlo ke ztrátě příznaku jiných přerušení, pokud by se jich sešlo více najednou.

Mezi další funkce hardware abstrahující vrstvy slouží detekce připojení a odpojení USB zařízení, uvedení sběrnice do a ze stavu resetu, spouštění a zastavování generování začátku rámce (SOF). Generování začátku rámce je důležité zejména v případě, kdy na sběrnici neprobíhá jiný přenos, toto generování začátků rámců je plně v režii hardwaru, pouze je třeba nastavit příslušným způsobem konfigurační registry modulu. Pokud by začátky rámců nebyly generovány, připojené zařízení by se uvedlo do stavu snížené spotřeby, kdy nereaguje na zasílané příkazy. Rovněž do této vrstvy patří funkce detekce přenosové rychlosti připojeného zařízení – zda se jedná o low-speed nebo full-speed zařízení, spolu s detekcí rychlosti připojeného zařízení úzce souvisejí i funkce pro nastavení registrů USB modulu pro korektní práci na odpovídající přenosové rychlosti. Posledně zmíněné funkce hrají důležitou roli v průběhu procesu enumerace.

Hojně využívanou funkcí hardware abstrahující vrstvy je generátor přerušení s periodou 1 ms. Tento generátor je využíván vrstvou USB host stacku jednak k odměřování času při čekání, jednak jako pojistka pro případ selhání při probíhajícím přenosu. Takto vzniklý časovač je přístupný jako funkce této vrstvy, nastavení časovače je provedeno pomocí funkce *usb_hal_wait()*, která má jako parametry číslo portu a čas v milisekundách. Po vypršení tohoto času dojde k signalizaci pomocí sdíleného příznaku, pokud není časovač pomocí *usb_hal_wait_stop()* zastaven dříve.

5.3 USB host stack

Jde o klíčovou vrstvu implementovaného systému, která obstarává většinu funkcí nezbytných pro komunikaci na sběrnici. Obsahuje funkci pro inicializaci host stacku, funkce pro obsluhu přenosu dat z vyšších vrstev aplikace a hlavní stavový stroj, který obstarává enumeraci zařízení po jeho připojení a samotné zasílání paketů při přenosech.

Inicializační funkce připravuje struktury v paměti do jasně definovaného stavu. Alokuje paměť pro struktury, kde je to nutné, nastavuje výchozí hodnoty a provázání jednotlivých struktur pomocí ukazatelů. Mezi inicializované struktury patří struktura stavu konkrétního USB portu, výchozí struktura pro připojené koncové zařízení a fronta požadavků na přenos. Dále tato funkce volá nízkoúrovňovou inicializaci hardware abstrahující vrstvy, která nastaví USB modul do výchozího stavu, ve kterém v případě připojení zařízení generuje přerušení a nastavuje sdílený příznak.

Funkce pro obsluhu přenosu dat z vyšších vrstev obsahuje funkce pro zahájení přenosu, funkce pro test dokončení operace a podobně. V principu jde vždy o funkci nastavující, resp. testující příznaky přenosu pomocí fronty požadavků, samotný přenos jako takový je realizován ve funkci hlavního stavového stroje.

Funkce hlavního stavového stroje – *usb_host_task()* se stará o obsluhu zařízení od jeho připojení přes enumeraci, veškeré datové přenosy až po jeho odpojení. Pro rovnoměrné rozložení zátěže je funkce realizována jako příkaz *switch* se základními stavy definovanými podle kapitoly 9 standardu USB 2.0[16]. V rámci některých z těchto základních stavů existují ještě podstavy, pokud je třeba rozčlenit prováděnou činnost do více samostatných operací. Například pro získání deskriptorů je třeba nejprve zaslat příkaz, poté přečíst data a na závěr potvrdit úspěšné přijetí dat, tedy pro tuto činnost jsou zpravidla definovány oddělené podstavy.

Při enumeraci je ze zařízení vyčten celý deskriptor zařízení a celý deskriptor konfigurace. Důležité informace z deskriptoru zařízení jsou uloženy ve struktuře popisující zařízení, konfigurační deskriptor je uložen v paměti a je možné na něj získat referenci pro jeho čtení aplikací pomocí vrstvy aplikačního rozhraní. Na základě deskriptorů je vytvořena provázaná hierarchie datových struktur k popisu jednotlivých rozhraní a jejich koncových bodů, včetně všech relevantních parametrů. Textové deskriptory nejsou získávány, protože jejich přítomnost není pro funkci systému důležitá, pokud by je však nadřazená aplikace potřebovala získat, je možné pomocí funkcí aplikačního rozhraní pro přenos dat tyto deskriptory získat dodatečně.

Všechny přenosy jsou zajištěny časovačem, který je implemenován v hardware abstrahující vrstvě. Pokud by tedy v průběhu nějakého přenosu došlo k chybě, nevyvolá tato chyba uváznutí, ale po vypršení časovače, které je signalizováno pomocí sdíleného příznaku, bude chyba detekována a následně provedeny nezbytné operace podle charakteru chyby, případně dojde k opakování přenosu. Časový interval časovače je závislý na druhu přenosu. Pokud je zasílán standardní příkaz, který není

následován datovou částí přenosu, je časovač podle standardu nastaven na 50 ms, pokud následuje datový přenos, je časovač nastaven na 500 ms, do kterých musí koncové zařízení zaslat odpověď. Opakování celého přenosu je při enumeraci možné pouze 3x, pokud by ani přesto nebyl přenos úspěšně dokončen, je další činnost pozastavena, protože se předpokládá porucha na zařízení, kterou lze odstranit pouze odpojením zařízení od hostitele. Při běžném přenosu vede třetí chyba k zablokování koncového bodu pro další použití a tento stav je třeba ošetřit zasláním příkazu *clear_feature*.

```
int usb_host_task(void)
 {
      int i,j;
      ...
      for(i = 0; i < USB_NUM_PORTS; i++)
      {
+           if(usb_hal_interrupt_flag[i] & DETACH_INTERRUPT) ...
            switch(port_descriptors[i].state.mainstate) {
+           case USB_DETACHED: ...
+           case USB_ATTAACHED: ...
+           case USB_POWERED: ...
+           case USB_DEFAULT: ...
+           case USB_ADDRESS: ...
+           case USB_CONFIGURED: ...
+           case USB_SUSPENDED: ...
+           case USB_HALTED: ...
            default:
                  led_red_on(); /* never should get here */
                  break;
            } /* end of switch */
      } /* end of for */
      return 0;
 }
```

Kód 5.3: Stavový stroj USB hostitele (zkráceno).

5.4 Vrstva aplikačního rozhraní

Vrstva aplikačního rozhraní zjednodušuje ovládání zasílání a čtení dat z aplikace, obsahuje funkce pro čtení datových struktur USB host stacku, které slouží například pro vyhledání konkrétního zařízení k získání jeho jednoznačné adresy podle třídy zařízení.

Funkce pro čtení a zápis zapouzdřují práci s daty po jednom znaku, čímž výrazně snižují režii a šetří přenosové pásmo na sběrnici. Zapouzdření čtení a zápisu probíhá obdobným způsobem jako u operačních systémů – při čtení je načteno více dat, pokud jsou dostupná, a po jednom znaku jsou předávána aplikaci. Při zápisu jsou data zapisována do bufferu a teprve po zaplnění bufferu nebo po vypršení časového intervalu dojde k jejich skutečnému odeslání na sběrnici. V případě, že je třeba odeslat data ihned, je přítomna funkce pro bezprostřední odeslání bufferu na sběrnici.

Kromě znakového přístupu k připojenému zařízení umožňuje čtení a zápis bloku dat blokujícím i neblokujícím způsobem. U neblokujícího způsobu je třeba testovat dokončení předchozí operace před provedením další operace. Funkce s blokujícím způsobem obsluhy navrací obsluhu volané

funkci teprve po dokončení, přičemž je možné pomocí návratové hodnoty testovat úspěšnost takové operace. Funkce s blokujícím způsobem obsluhy iniciuje přenos a opakovaně volá funkci *usb_host_task()* pro zpracování tohoto požadavku, průběžně přitom testuje stav prováděné operace a teprve po jejím dokončení navrací řízení volající aplikaci.

Výrazné zjednodušení při vytváření aplikace umožňuje funkce aplikačního rozhraní *usb_api_find_class_device()* viz Kód 5.4, která prohledá dostupné USB porty hostitele, zařízení připojená k těmto portům a mezi těmito zařízeními hledá zařízení, které patří do třídy předané této funkci parametrem, je-li takové zařízení nalezeno, je návratovou hodnotou jednoznačná adresa tohoto zařízení. Programátor aplikace se tak nemusí zajímat o to, kolik portů má zařízení hostitele, ke kterým z těchto portů je připojeno nějaké zařízení a ručně zjišťovat parametry takového zařízení.

```
int usb_api_find_class_device(BYTE dev_class)
{
      int i;
      DEVICE_INFO *dev_ptr;
      for(i = 0; i < usb_api_get_num_ports(); i++)
      { /* scan for devices over all ports */
            dev_ptr = port_descriptors[i].device;
            while(dev_ptr != NULL)
            {
                  if(dev_ptr->class == dev_class &&
port_descriptors[i].state.mainstate = USB_CONFIGURED)
                        return ((i <<8) | (dev_ptr->address));
                  dev_ptr = dev_ptr->next;
            }
      }
      return 0;
}
```

Kód 5.4: Funkce pro vyhledání zařízení dle jeho třídy.

Obdobná funkce jako pro vyhledání zařízení je i pro prohledání rozhraní (skupin koncových bodů, viz podkapitola 2.3.3) v rámci zvolené konfigurace konkrétního zařízení určeného jeho adresou. V tomto případě je navrácen index rozhraní. Pomocí adresy zařízení a indexu rozhraní lze následně zjistit další parametry tohoto rozhraní, například protokol rozhraní nebo počet koncových bodů nebo čísla koncových bodů rozhraní. S využitím adresy zařízení a čísla koncových bodů je možná komunikace mezi aplikací a koncovým zařízením pomocí dříve zmíněných funkcí pro zasílání a čtení bloků dat, případně čtení a zápis jednotlivých znaků s ukládáním do vyrovnávacích bufferů.

5.5 Vrstva demonstrační aplikace

Demonstrační aplikace ke svému provádění využívá nižších vrstev, zejména vrstvy aplikačního rozhraní. Aplikace je pro správnou funkci bez plánovače úloh řešena jako stavový stroj, ve kterém je kód rozčleněn do krátkých úseků. Stavový stroj je realizován pomocí příkazu switch, obdobně jako ve vrstvě USB host stacku, viz Kód 5.5.

Aplikace s využitím funkcí aplikačního rozhraní testuje, zda je k hostiteli připojeno nějaké zařízení, pokud ano, hledá mezi připojenými zařízeními koncové zařízení třídy CDC. Pokud je zařízení třídy CDC nalezeno, jsou vyčteny čísla koncových bodů komunikačního a datového rozhraní.

Pokud jsou čísla koncových bodů získána, je nastaven příznak ovládání zařízení touto aplikací. Tento příznak slouží pro vrstvu USB hostitele, v podstatě znamená načtení ovladače pro zařízení. Příznak je nezbytný z více důvodů. Jednak slouží aplikaci pro detekci odpojení zařízení, jednak zabraňuje vícenásobnému použití téhož zařízení více aplikačními úlohami zároveň, zabraňuje se tak konfliktům. Po úspěšném provedení těchto kroků je zařízení aplikací konfigurováno pro požadovanou funkci, například při použití převodníku USB→RS-232 je nastavena odpovídající přenosová rychlost, počet datových a stop bitů, parita. Po konfiguraci zařízení přejde aplikace do režimu komunikace.

```
 void usb_application_task(void) /* main application task */
{
  ...
  if(device_handle != NULL && device_handle->state == USB_DETACHED)
  { /* check whether handled device is still connected */
    ...
  }
  switch(usb_application_state) { /* state machine */
  case USB_APPL_STATE_NO_DEVICE: /* search for device */
    address = usb_api_find_class_device(CDC_CLASS);
    if(address != 0 && usb_api_is_device_configured(address))
      usb_application_state = USB_APPL_DEVICE_NOT_SET;
    break;
  case USB_APPL_DEVICE_NOT_SET: { /* get endpoint numbers */
    ...
    temp = usb_api_find_class_interface(address, CDC_DATA_CLASS);
    data_out_ep = usb_api_get_ep_num(address, (BYTE)temp, 0);
    if(usb_api_get_ep_dir(address, data_out_ep) == DIR_OUT)
      data_in_ep = usb_api_get_ep_num(address, (BYTE)temp, 1);
    else ...
    break;
+ case USB_APPL_CONFIGURE_DEVICE: ...
  case USB_APPL_DEVICE_CONFIGURED: /* process data */
    temp = usb_api_read(address, data_in_ep, app_buffer, 8);
    for(i = 0; i < temp; i++)
    {
      if(app_buffer[i] >= 'A' && app_buffer[i] <= 'Z')
        app_buffer[i] += 'a' - 'A';
      ...
    }
    if(temp != 0)
      usb_api_write(address, data_out_ep, app_buffer, temp);
    break;
  default:
    led_red_on(); /* never should get here */
    break;
  }
  return;
}
```

Kód 5.5: Hlavní úloha aplikační vrstvy.

V režimu komunikace je v demonstrační aplikaci prováděna konverze přijatých písmen na opačnou velikost. Malá písmena na velká a velká na malá a konvertovaná data jsou přeposílána zpět

do koncového zařízení. Čtení a zápis dat probíhá s využitím vrstvy aplikačního rozhraní. Průběžně probíhá testování, zda nedošlo k odpojení používaného zařízení, aby se aplikace nedostala do nedefinovaného stavu použitím struktur v paměti, které byly mezitím díky odpojení zařízení uvolněny.

Při studiu příkazů zařízení třídy CDC jsem čerpal především z [17]. V demonstrační aplikaci je vzhledem k problémům zmíněných v kapitole 6 (nedostupnost širšího spektra zařízení) využito pouze omezené sady příkazů. Tyto příkazy se omezují na nastavení a kontrolu parametrů linky, konkrétně jde o příkazy *set_line_coding*, *get_line_coding* a *set_line_state*, obdobných příkazů by bylo použito například při obsluze řízení modemu nebo telefonu.

Příkaz *get_line_coding* slouží k zjištění současného nastavení linky. Konkrétně jde o informaci o přenosové rychlosti, počtu datových bitů, počet stop bitů a zvolená parita přenášených dat. Příkaz *set_line_coding* nastavuje výše zmíněné parametry. Aplikace nejdříve zjistí současné nastavení linky pomocí *get_line_coding*, pokud se nastavení odlišuje od požadovaného nastavení, zašle příkaz *set_line_coding* a následně opět zjišťuje, zda došlo k přenastavení parametrů na požadované hodnoty. Příkaz *set_line_state* slouží k nastavení pomocných signálů rozrhaní RS-232, konkrétně nastavení hodnot na vodičích DTS, DTR. Příkaz zde je především z demonstračních účelů, vývojová deska Atmel příkaz sice podporuje, k reálnému nastavení hardwaru však neslouží.

6 Testování a zhodnocení výsledků

Kapitola popisuje možné způsoby testování, kterých lze využít při vývoji aplikace pro USB, shrnuje poznatky získané při vývoji implementovaného systému. Ve druhé části kapitoly jsou zhodnoceny dosažené výsledky práce a popsána možná rozšíření implementované aplikace.

6.1 Testování

Testování může probíhat na různých úrovních. V úvahu připadá testování na fyzické úrovni, testování na úrovni USB protokolu a testování na aplikační úrovni. K testování na fyzické úrovni je potřeba hardwarový USB analyzátor. Hardwarové analyzátory pro USB sběrnici slouží jednak ke kontrole napěťových úrovní a časování generovaných zařízením, tak zpravidla pro nízkoúrovňovou kontrolu USB protokolu, včetně přenosů zatížených různými chybami (chyba kontrolního součtu, ...), které se při testování na vyšších vrstvách objeví pouze příznakem „došlo k chybě“ nebo se neobjeví vůbec. Tyto analyzátory zpravidla neslouží pouze k monitorování provozu na sběrnici, ale často jsou vybaveny i funkcemi pro generování provozu na sběrnici, je tak možný samostatný vývoj koncového zařízení bez hostitele nebo vývoj hostitele bez koncového zařízení. Výhodou těchto analyzátorů jsou široké možnosti testování, kdy jsou zobrazeny všechny informace, které se na fyzickém spoji objeví, a nezávislost na operačním systému, nevýhodou je potom cena, která začíná přibližně na 10 tisících Kč u nejnižších modelů (pouze full-speed zařízení, bez generování provozu).

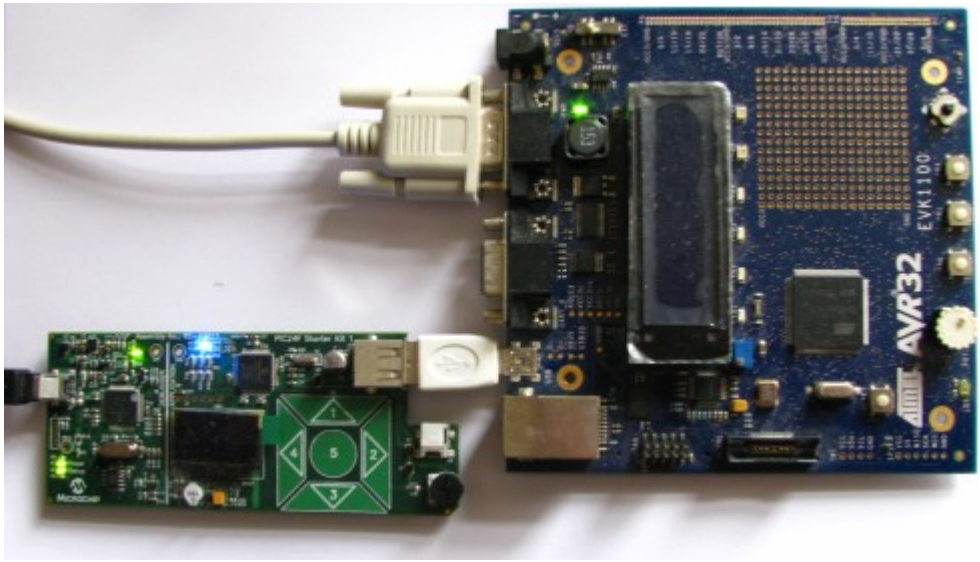
Další možností testování jsou softwarové analyzátory. Zpravidla se jedná o počítačový program, který přebírá informace z ovladače kořenového hostitele v počítači. Často jsou dostupné i zkušební verze s omezenými funkcemi (např. omezená délka záznamu) nebo s časově omezenou funkčností. Tyto programy většinou slouží pouze k monitorování provozu mezi hostitelem v osobním počítači a koncovým zařízením, dokonce některé ignorují pakety doručené s chybou, pokud je chyba detekována, je dostupný pouze příznak chyby, nikoliv přímo její příčina. Z výše zmíněných důvodů je patrné, že je možné tyto programy použít především pro odladění problémů s deskriptory vyvíjeného koncového zařízení, pro ladění aplikace hostitele ve vestavěném zařízení jsou tyto analyzátory zcela nevhodné. Výhodou softwarových analyzátorů je jejich nižší cena, která je například u programů USBTrace [18] nebo USBlyzer [19] 200 $ za neomezenou licenci pro jednoho uživatele, v přepočtu tedy kolem 4 tisíc Kč, nevýhodou těchto analyzátorů je výrazně omezená funkčnost. Ze zmíněných programů se mi USBTrace jeví jako výrazně lepší volba.

Testování implementované aplikace bylo výrazně omezeno nedostupností vhodných prostředků. Hardwarový analyzátor se mi bohužel nepodařilo zapůjčit nebo k němu alespoň získat přístup prezenční formou. Softwarový analyzátor vzhledem k zmíněným nedostatkům rovněž nešel použít pro ladění samotné aplikace, přesto jsem softwarového analyzátoru použil alespoň k ověření správnosti načtených deskriptorů vyvíjeným USB hostitelem, kdy jsem porovnal informace o koncovém zařízení ze softwarového analyzátoru a z USB hostitele. Ladění aplikace s využitím vývojového prostředí rovněž nebylo možné z důvodů zmíněných v podkapitole 3.1.2.

Testování na úrovni USB protokolu jsem původně chtěl provádět s využitím vývojového kitu pro Atmel AVR, který jsem měl zapůjčen. Pomocí tohoto kitu jsem chtěl simulovat sériové zařízení třídy CDC a probíhající komunikaci na sběrnici USB pomocí jiného rozhraní (UART, případně Ethernet) přeposílat do terminálu v osobním počítači. Tímto způsobem testování jsem chtěl zejména

ověřit správný průběh enumerace, časování a sled paketů a následně i tvar příkazů AT zasílaných aplikací v hostitelském systému koncovému sériovému zařízení. Bohužel jsem přístup k vývojovému kitu Atmel získal opět až po dokončení implmentace enumerace a souvisejícího kódu. Díky důslednému testování návratových hodnot, stavu dokončených přenosů, pomocných úseků kódu a vyčtení paměti kontroléru ve vývojovém prostředí MPLAB bylo možné otestovat průběh enumerace i podpůrných prostředků i bez analyzátoru, byť to znamenalo značné zpoždění proti původnímu plánu. V této fázi testování jsem se potýkal s problémy, kdy nebylo zřejmé, proč aplikace neodpovídá předpokládaným způsobem. Bylo proto třeba doplnit podpůrný kód k identifikaci chyby, který v některých případech objemem výrazně převyšoval kód samotné aplikace. Tyto problémy se nakonec podařilo odstranit. Ve většině případů se jednalo o programátorské chyby, které by bylo možné snadno identifikovat v případě dostupného analyzátoru nebo jiného rozhraní mikrokontroléru, na vývojovém kitu však není dostupné ani sériové rozhraní, proto si vyhledání těchto chyb vyžádalo hodně času a jednorázově použitého kódu.

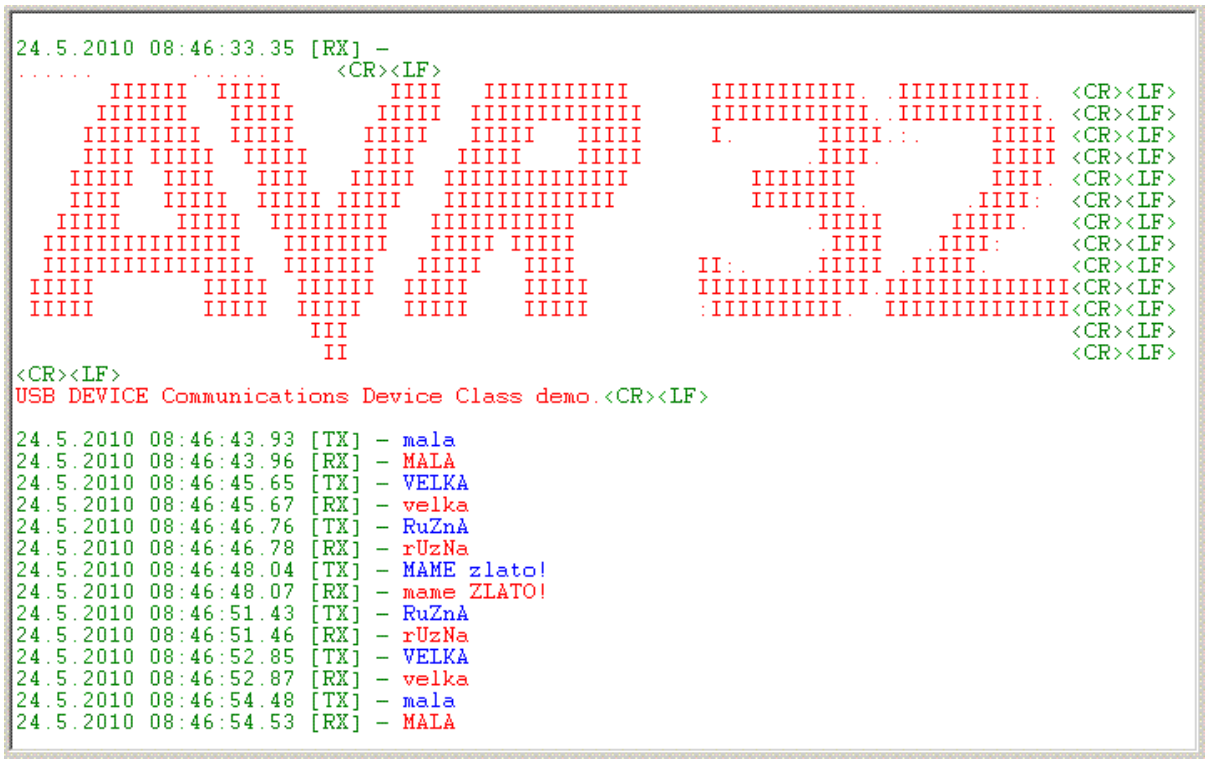
Na aplikační úrovni probíhalo testování na reálném koncovém zařízení. Původně jsem chtěl využít dedikovaného převodníku USB → RS-232, případně USB modemu nebo GSM telefonu, bohužel všechna taková zařízení, která vlastním nebo se mi je podařilo zapůjčit nebyly třídy CDC, ale využívaly protokolu specifického pro jednotlivé výrobce nebo spíš konkrétní výrobky těchto výrobců. Tyto specifické protokoly většinou nejsou veřejně dokumentovány, veškerá dokumentace se odkazje na dodaný ovladač pro běžně rozšířené operační systémy pro osobní počítače, a reverzní inženýrství protokolu s využitím protokolových analyzátorů je výrazně nad rámec této práce. Jedinou možností tedy zůstalo využití vývojového kitu Atmel, do kterého jsem nahrál firmware zařízení třídy CDC. Vývojový kit tak slouží jako převodník USB → RS-232. Jako hostitel tohoto převodníku je použit Microchip Starter Kit s implementovaným systémem, jako druhý konec převodníku je osobní počítač se spuštěným terminálem monitorujícím sériový port. Sestava vývojových prostředků využitých pro testování viz Obrázek 6.1.



Obrázek 6.1: Sestava vývojových přípravků při testování.Vlevo MPLAB Starter Kit, vpravo vývojový kit Atmel EVK1100.

Po spuštění sestavy došlo k detekci připojeného USB zařízení vrstvou USB hostitele. Dokončení enumerace ze strany hostitele je signalizováno rozsvícením modré diody na Starter Kitu.

Na straně koncového zařízení je po inicializaci mikrokontroléru na kitu v terminálu vypsán obrázek pomocí ascii znaků, doknočení enumerace kitu Atmel je potom signalizováno hláškou „USB DEVICE Communications Device Class Demo.“, viz Obrázek 6.2. Na obrázku je též vidět odesílaná data z počítače do zařízení Atmel, která jsou přeposlána USB hostiteli (modře), data zpracovaná hostitelem (písmena převedená na opačnou velikost) jsou přeposlána zpátky do zařízení a následně do terminálu (červeně dole).



Obrázek 6.2: Výpis terminálu při testování.

6.2 Zhodnocení práce

Jsem rád, že se i přes potíže podařilo vytvořit funkční aplikaci s alternativním přístupem k řešení s minimalizovanou obsluhou v rutině přerušení a bez plánování spuštěných úloh na kontroléru. Funkčnost celého systému byla ověřena s využitím dalších dostupných prostředků.

Z problémů, které bylo třeba řešit v průběhu implementace, pro mě plyne ponaučení, že na demonstračních vývojových kitech jako je MPLAB Starter Kit je lepší využívat demonstračních příkladů a tyto v omezeném rozsahu modifikovat. Pro vývoj aplikace od úplného začátku bych rozhodně doporučil použití jiných vývojových prostředků s širšími možnostmi pro ladění, například vývojový kit Microchip Explorer 16 s použitím rozšiřujícího modulu pro USB, viz [20], ideálně doplněné o hadrwarový USB analyzátor, poté by byl vývoj výrazně rychlejší.

Možná rozšíření spočívají ve třech oblastech. První oblastí je hardware abstrahující vrstva, kterou je možno rozšířit o podporu tzv. ping-pong módu (též se používá označení double-banking), kdy je využito dvou alternativních vyrovnávacích bufferů pro každý koncový bod. Toto rozšíření slouží především pro zvýšení propustnosti v aplikacích s pevně danými adresami vyrovnávací paměti USB modulu. U mikrokontrolérů rodiny PIC24F toto chování není tak důležité, protože USB modul

využívá přímého přístupu do paměti a před jednotlivým přenosem je potřeba nastavit pouze ukazatel na paměť využitou pro přenos. Z výše uvedených důvodů je tedy možné, že režie obsluhy přepínání jednotlivých bufferů by převýšila režii spojenou s nastavením jiného ukazatele a zvýšení propustnosti by se tak nekonalo. Tato úvaha platí pro implementované řešení s minimalizovanou obsluhou přímo v rutině obsluhy přerušení. Zmínka o problematice vyrovnávacích bufferů je např. na straně 143-144 v [21].

Druhou oblastí pro možné rozšíření je podpora izochronních přenosů, která nebyla implementována. S izochronními přenosy by bylo třeba řešit jiný způsob iniciace přenosů a dokončení přenosů, protože izochronní komunikace není potvrzována. Dále by bylo třeba zajistit řízení vytížení šířky pásma sběrnice při podpoře více současně připojených zařízení k jednomu portu.

Třetí oblastí rozšíření je implementace ovladačů zařízení jiných tříd s využitím vytvořeného aplikačního rozhraní. V současné době je nad aplikačním rozhraním implementována pouze demonstrační aplikace, která spolupracuje se zařízeními třídy CDC s asynchronním režimem činnosti (např. již zmíněný převodník USB → RS-232).

7 Závěr

Cílem práce bylo vytvoření USB hostitele na mikrokontroléru PIC24F, který bude schopen obsluhovat sériové zařízení třídy CDC. Tohoto cíle bylo dosaženo, byl vytvořen vícevrstvý systém USB hostitele s aplikací využívající aplikačního rozhraní. Řešení je až na hardware abstrahující vrstvu přenositelné na úrovni zdrojových kódů.

Doufám, že mnou vytvořená aplikace poslouží jiným jako výchozí bod pro jejich aplikaci, případně poslouží jako inspirace svým alternativním přístupem k problému. Implementované řešení by mělo zaujmout přenositelností a snadnou rozšiřitelností systému. Alternativní přístup, tedy aplikace hostitele s minimalizovanou obsluhou v rámci obslužné rutiny přerušení, vidím jako můj přínos do problematiky.

Možnosti dalšího vývoje aplikace byly popsány v podkapitole 6.2, proto uvedu jen jejich stručné shrnutí. Možné rozšíření spočívá v implementaci podpory ping-pong módu (rozšíření hardware abstrahující vrstvy), podpory izochronních přenosů (rozšíření vrstvy USB host stacku) a podpory jiných tříd zařízení s využitím vytvořeného aplikačního rozhraní (rozšíření aplikační vrstvy).

Osobní přínos práce pro mě spočívá ve zkušenostech s implementací rozsáhlejší aplikace, které není možné nasbírat při práci na výrazně menších projektech, jež jsou v rámci jednotlivých předmětů na fakultě prováděny. S větším rozsahem projektu je třeba důsledně rozmýšlet souvislosti jednotlivých částí celého systému, u systémů implementovaných v mikrokontroléru bez operačního systému to platí dvojnásob. Rovněž získané zkušenosti jasně naznačují, že je vhodné zajistit si vývojové prostředky odpovídající rozsahu a povaze implementace. Potvrdilo se také, že kvalitní návrh je klíčem k úspěchu, proto v budoucích projektech budu klást na návrh ještě větší důraz, bohatě se to vyplatí při implementaci.

Nasazení implementovaného systému je možné v celé řadě aplikací, například jako komunikační rozhraní vestavěných aplikací, kdy je využito hotového zařízení (například modem) připojeného pomocí USB, odpadá tak analýza, návrh a vývoj tohoto rozhraní, vývoj celku je tak mnohem kratší a levnější. Dalším využitím je servisní rozhraní specializovaných zařízení nebo elektroniky obecně, kde je možné pomocí USB rozhraní a flash disku získávat ze zařízení záznamy o provozu, chybová hlášení nebo poskytovat možnost aktualizace firmwaru, který nějakým způsobem vylepšuje vlastnosti takového zařízení bez potřeby fyzického zásahu uvnitř zařízení.

Jsem si jist, že zkušenosti získané při analýze, návrhu a implementaci problému řešeného v rámci diplomové práce v budoucí praxi bohatě využiji a přetavím je ve svůj prospěch.

Literatura

[1] GOOK, Michael. *PC hardware interfaces - A Developer's Reference*. Wayne, Pennsylvania: A-LIST Publishing, 2004. 672s. ISBN 1-931769-29-X.

[2] *JimPrice.Com* [online]. 2009 [cit. 2009-01-03]. Nullmodem.Com. Dostupné z WWW: <http://nullmodem.com/>.

[3] *Wikimendia Foundation* [online]. 2009 [cit. 2009-01-03]. FireWire. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Firewire>.

[4] *CKSinfo.com* [online]. 2009 [cit. 2009-01-03]. Free Connectors Clipart Images, Graphics, Animated Gifs & Animations. Dostupné z WWW: <http://www.cksinfo.com/electronics/computers/connectors/index.html>.

[5] *The Serial ATA International Organization* [online]. 2009 [cit. 2009-01-03]. eSATA. Dostupné z WWW: <http://www.sata-io.org/technology/esata.asp>.

[6] *Computer Solutions Ltd* [online]. 2009 [cit. 2009-12-06]. Embedded USB - a brief tutorial. Dostupné z WWW: <http://www.computer-solutions.co.uk/info/Embedded_tutorials/usb_tutorial.htm>.

[7] *Embedded.com* [online]. 2009 [cit. 2009-01-03]. Internet Appliance Design: An Introduction to USB Development. Dostupné z WWW: <http://www.embedded.com/2000/0003/0003ia2.htm>.

[8] *USB Implementers Forum, Inc.* [online]. 2009 [cit. 2009-12-06]. About USB Implementers Forum, Inc.. Dostupné z WWW: <http://www.usb.org/about>.

[9] *3dnews.ru* [online]. 2009 [cit. 2009-01-03]. Опубликованы спецификации USB 3.0. Dostupné z WWW: <http://www.3dnews.ru/tags/передача>.

[10] *geekstreamonline.com* [online]. 2009 [cit. 2009-01-03]. Geekstream converage of the 2008 International CES, Las Vegas NV. Dostupné z WWW: <http://www.geekstreamonline.com/CES.html>.

[11] *On-The-Go and Embedded Host Supplement to the USB Revision 2.0 Specification*. S.l.: USB Implementers Forum, Inc., 8.5.2009. 79s. Dostupné z WWW: <http://www.usb.org/developers/onthego/USB_OTG_and_EH_2-0.pdf>.

[12] *Requirements and Recommendations for USB Products with Embedded Hosts and/or Multiple Receptacles*. S.l.: USB Implementers Forum, Inc., 8.7.2004. 18s. Dostupné z WWW: <http://www.usb.org/developers/docs/EH_MR_rev1.pdf>.

[13] *Beyond Logic* [online]. 2009 [cit. 2009-01-05]. USB in a NutShell: Making sense of the USB standard. Dostupné z WWW: <http://www.beyondlogic.org/usbnutshell/>.

[14] ANDERSON, Don. *USB system architecture (USB 2.0)*. Colorado Springs, Colorado: MindShare, Inc., 2001. 543s. ISBN 0-201-46137-4.
[15] *Transfer Multisort Elektronik Sp. z o.o.* [online]. 2010 [cit. 2010-05-15]. MICROCHIP development kits MPLAB® Starter Kit for PIC24F. Dostupné z WWW: <http://www.tme.eu/html/EN/microchip-development-kits-mplab__-starter-kit-for-pic24f/ramka_6878_EN_pelny.html>.
[16] *Universal Serial Bus Revision 2.0*. S.l.: USB Implementers Forum, Inc., 27.5.2000. 650s. Dostupné z WWW: <http://www.usb.org/developers/docs/usb_20_050710.zip>.
[17] *USB Class Definitions for Communication Devices*. S.l.: USB Implementers Forum, Inc., 19.1.1999. 121s. Dostupné z WWW: <http://www.usb.org/developers/devclass_docs/usbcdc11.pdf>.
[18] *SysNucleus* [online]. 2010 [cit. 2010-05-19]. USBTrace : Software-only USB Protocol Analyzer. Dostupné z WWW: <http://www.sysnucleus.com/>.
[19] *usblyzer.com* [online]. 2010 [cit. 2010-05-19]. USBlyzer - Professional Software USB Protocol Analyzer. Dostupné z WWW: <http://www.usblyzer.com/>.
[20] *HW server s.r.o.* [online]. 2010 [cit. 2010-05-19]. MPLAB ICD 3 + Explorer 16 Kit. Dostupné z WWW: <http://obchod.hw.cz/Default.asp?cls=stoitem&stiid=42097>.
[21] AXELSON, Jan. *USB Complete*. Madison, WI: Lakeview Research LLC, 2005. 593s. ISBN 1-931448-03-5.

Seznam příloh

Příloha 1. CD s elektronickou verzí textu práce, zdrojovými kódy a projektem prostředí MPLAB.