UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2009 Jiří Paar

UNIVERZITA PARDUBICE

FAKULTA ELEKROTECHNIKY A INFORMATIKY

Rozhraní USB v řídících aplikacích

Jiří Paar

Bakalářská práce

2009

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra elektrotechniky
Akademický rok: **2008/2009**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jiří PAAR**

Studijní program: **B2612 Elektrotechnika a informatika**

Studijní obor: **Komunikační a mikroprocesorová technika**

Název tématu: **Rozhraní USB v řídících aplikacích**

Zásady pro vypracování:

Proveďte detailní rozbor principů a vlastností rozhraní USB s o hledem na jeho aplikaci ve vestavěných zařízeních. Pro praktické testy navrhněte jednoduchý vývojový kit s mikrokontrolérem s jádrem Atmel AVR s vestavěným USB řadičem. Funkčnost zařízení ověřte stavbou vzorku. Pro vývojový kit naprogramujte firmware, s jehož pomocí bude možné demostrovat vlastnosti různých možností spolupráce mikrokontroléru s USB rozhraním a řídícího PC. Osnova práce: Rozhraní USB - elektrické vlastnosti, komunikační model. Současné možnosti využití rozhraní USB ve vestavěných aplikacích - přehled dostupné součástkové základny. Programátorský model řadiče zvoleného mikrokontroléru Návrh HW vývojového kitu. Programování firmware a testy zařízení.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

Specifikace USB 2.0: Dostupné online na http://www.usb.org
Atmel AVR USB Software Packages: Dostupné online na http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4441
USB popis rozhraní:
Dostupné online na
http://hw.cz/Teorie-a-praxe/Dokumentace/ART327-USB—Universal-Serial-Bus—Popis-rozhrani.html

Vedoucí bakalářské práce: **Ing. Martin Hájek**
Katedra elektrotechniky

Datum zadání bakalářské práce: **15. ledna 2009**

Termín odevzdání bakalářské práce: **15. května 2009**

doc. Ing. Simeon Karamazov, Dr.
děkan

L.S.

Ing. Zdeněk Němec, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2009

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezentačním zpřístupněním své práce v Univerzitní knihovně Univerzity Pardubice.

V Pardubicích dne 24. 8. 2009

Jiří Paar.

Poděkování:

Na úvod bych rád poděkoval vedoucímu práce panu ing. Martinovi Hájkovi za odborné vedení a rady v průběhu práce, také za možnost výroby vývojového kitu. Také bych mu rád poděkoval za poskytnutí možnosti spolupráce na projektu pro společnost Steinel Technik, kde se výsledky práce mohli ověřit v praxi. Poděkování také patří všem, kdo mě v práci pomáhali a podporovali.

SOUHRN

Tato práce se zabývá způsoby realizace komunikace přes USB sběrnici ve vestavěných aplikacích. Práce je zaměřena na praktickou práci s mikrokontrolérem AT90USB647 s implementovaným USB řadičem. Součástí práce je stavba vývojového kitu s vybraným mikrokontrolérem, na němž bylo naprogramováno a úspěšně otestováno několik verzí firmwaru (např. třídy CDC a HID).

KLÍČOVÁ SLOVA

USB sběrnice; USB řadič; mikrokontrolér

TITLE

USB interface in embedded applications

ABSTRACT

This work deals with the ways of implementation of communication over USB in embedded applications. Work is focused on practical work with AT90USB647 microcontroller with embedded USB controller. Development kit with selected MCU was built and several versions of application firmware (CDC and HID class implementation) were programmed and successfully tested.

KEYWORDS

USB bus; USB controller; microcontroller

Seznam zkratek:

CDC	Comunnication Devices Class	Třída pro komunikační zařízení
CRC	Cyclic Redundancy Check	Cyklický redundantní kód
DRRAM	Double Port Random Access Memory	Dvou banková paměť.
FIFO	First In First Out	Datová struktura typu fronta.
FTDI	Future Technology Devices International	Název společnosti.
HID	Human Interface Device	Třída pro zařízení určená ke komunikaci s uživatelem.
HNP	Host Negation Protocol	Protokol pro záměnu typu zařízení.
MPSSE	Multi-Protocol Synchronous Serial Engine Interface	Rozhraní pro tvorbu sériových sběrnic.
OTG	On-The-Go	Nadstavba zařízení USB device.
PID	Packet ID	Identifikátor paketu.
PID	Product ID	Identifikátor produktu.
SRP	Session Request Protocol	Protokol pro vzdálené probuzení zařízení USB host.
UART	Universal Asynchronous serial Reciever and Trasmitter	Zařízení umožňující příjem i vysílání asynchronních sériových dat.
USB	Universal Serial Bus	Univerzální sériová sběrnice.
SOF	Start Of Packet	Typ rámce obsahující číslo daného rámce.
VID	Vendor ID	Identifikátor výrobce

Úvod

USB sběrnice se postupem času, kdy prošla jistým vývojem, stala nejpoužívanějším rozhraním pro připojení zařízení k počítači. Dnes již neexistuje snad žádné zařízení, které by nebylo, nebo nemohlo být, vybaveno touto sběrnicí, proto se stala pomalu běžnou součástí každodenního života. Následně začala vytlačovat jiná rozhraní do té doby běžná pro stolní počítače, jako je např. sériové rozhraní RS232 nebo paralelní port. Jejich velkou výhodou byla jednoduchost použití, proto byla často používaná ve vestavěných aplikacích, zejména pak rozhraní RS232. U mnohých přenosných počítačů již tato rozhraní nenajdeme. Byla nahrazena především kvůli jejich nízké přenosové rychlosti a poměrně velkému počtu vodičů. Vývojáři vestavěných aplikací tak byli nuceni k přechodu na USB sběrnici. Přechod umožnili specializované obvody v podobě převodníků, např. USB sběrnice na RS232, nebo univerzální mikrokontroléry s USB řadičem. Obě možnosti se liší ve způsobu začlenění do výsledné aplikace a možnostech samotného obvodu.

Text bakalářské práce je zaměřen na popis principů USB sběrnice pro použití ve vestavěných zařízeních. Popsané principy jsou využity pro pozdější sestavení funkčních zařízení využívajících integrovaný USB řadič mikrokontorléru. Zároveň je popsán princip nastavování, řízení a kontroly systému použitého řadiče. Součástí práce je také shrnutí dostupných integrovaných obvodů pro použití USB sběrnice ve vestavěných aplikacích.

V první kapitole je popsán základní princip USB sběrnice, její topologie, používané konektory, verze sběrnice, popis základních typů paketů, základní třídy USB zařízení a základní typy deskriptorů. Druhá kapitola popisuje dnes dostupné obvody pro vestavěná zařízení, která zprostředkovávají přístup k USB sběrnici. Třetí kapitola je zaměřena na popis USB řadiče vybraného mikrokontroléru. Ve čtvrté kapitole lze najít popis vývojového kitu s popsaným mikrokontrolérem ze třetí kapitoly. Závěrečná kapitola se věnuje popisu jednoho z vyvinutých obslužných firmwarů. Vybraným firmwarem je virtuální sériový port.

Obsah:

Seznam obrázků:

Seznam tabulek:

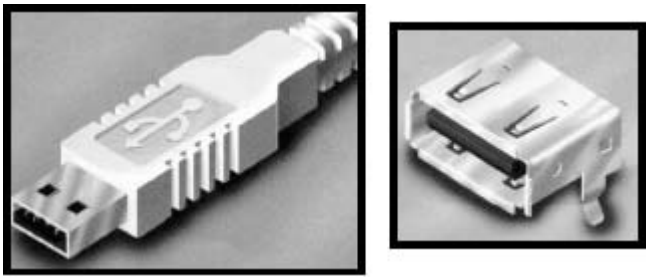
1 Základní popis USB sběrnice

Tato kapitola se zaměří na popis základních vlastností USB (Universal Serial Bus) sběrnice a zejména na sběrnici verze USB 1.1. Která je pro vestavěné aplikace svými rychlostmi spíše dostupná než verze USB 2.0. Na USB sběrnici se většinou vyskytují dva typy zařízení (USB host a USB device). Veškeré informace o USB sběrnici lze nalézt v [1].

USB sběrnice používá pro propojení zařízení celkem čtyři vodiče (VBUS, GND, D+, D–). Vodiče VBUS a GND slouží k napájení připojeného zařízení, toto napájení může, ale také nemusí, být využito. Toto napájení je velikou předností USB sběrnice. Poslední dva vodiče (D+ a D–) jsou datové vodiče, které tvoří diferenciální sběrnici. Tyto dva vodiče slouží k obousměrné komunikaci, proto je komunikace pouze typu halfduplex. Hodinový signál se nepřenáší pomocí žádného speciálního vodiče, proto musí být synchronizace vysílací a přijímací strany zajištěna jinými prostředky.

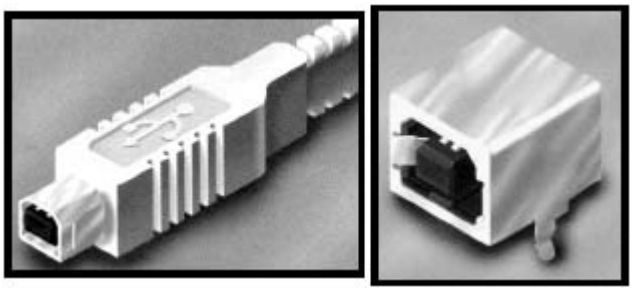
1.1 Typy USB konektorů

Pro připojení USB zařízení slouží několik typů konektorů. Základními dvěma typy konektorů jsou konektory A a B.



Obrázek 1 Konektor typu A [2]

Konektor typu A je určen pro zařízení typu USB host.



Obrázek 2 Konektor typu B [2]

Konektor typu B je výhradně určen pro zařízení typu USB device.

Pro velké rozměry těchto konektorů byly přidány další typy konektorů, které jsou již výrazně menší a svými rozměry vyhovují pro dnešní zařízení. Tímto novým konektorem je typ mini-B, který je určen pro zařízení USB device.

Obrázek 3 Konektor typu mini-B [2]

1.2 Topologie USB sběrnice

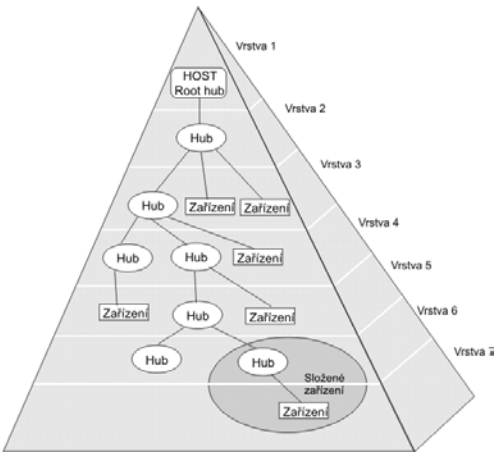
USB sběrnici je možné zapojit jen podle určitých pravidel, proto je možné definovat určitou topologii zapojení zařízení na sběrnici. Topologii lze rozdělit na dvě samostatné části. Jednou částí je fyzická topologie, tedy to, jak zařízení ke sběrnici připojit. Druhá část je tvořena logickou topologií. Tím je myšleno, jak jednotlivá zařízení spolu komunikují.

1.2.1 Fyzická topologie

Fyzickou topologií se rozumí skutečné propojení jednotlivých zařízení na USB sběrnici. Na USB sběrnici mohou být přítomny pouze tři typy zařízení. Základním typem zařízení, které musí být přítomno na každé USB sběrnici, je zařízení typu USB host. Toto zařízení představuje hlavní řídící člen starající se o veškerou komunikaci a musí být připojeno ke sběrnici pouze jedno. Dalším typem, bez kterého by nemohla být tvořena USB sběrnice, je zařízení typu USB device, tj. členem, se kte-

rým USB host komunikuje. Těchto zařízení může být na sběrnici až 127. Posledním nepovinným typem zařízení je USB HUB, který pouze rozšiřuje počet portů zařízení USB host.

Fyzickou topologii si lze představit podle následujícího schématu:



Obrázek 4 Ukázka fyzické topologie USB sběrnice [3]

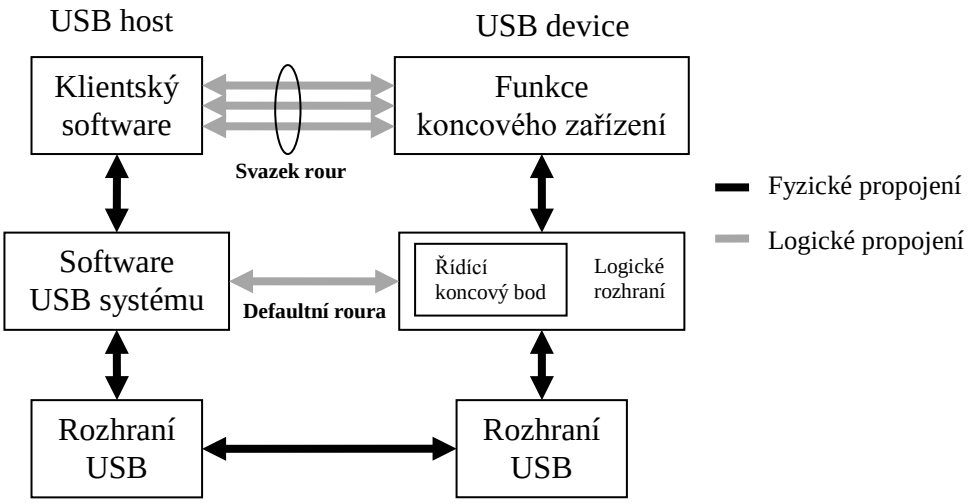
Zařízení USB host je většinou tvořeno klasickým počítačem, protože musí obstarat celé řízení sběrnice. Mezi kořenovým zařízením USB host a koncovým zařízením USB device může být maximálně pět zařízení typu USB HUB. V počátcích USB sběrnice nebyla jiná možnost, jak připojit jiné USB zařízení. Postupem času se začalo ukazovat, že velkou nevýhodou sběrnice je nutnost přítomnosti zařízení USB host. Proto vzniklo rozšíření o další typ zařízení, resp. o nadstavbu zařízení USB device. Tato nadstavba se nazývá OTG (On-The-Go) režim, popis tohoto režimu lze nalézt v [4] nebo [2].

OTG režim dovoluje, aby zařízení USB device fungovalo jako zařízení USB host. Tak je např. možné propojit digitální fotoaparát přímo s USB tiskárnou a tisknout fotografie, nebo propojit dva mobilní telefony, což jsou typické zařízení typu USB device. Např. v případě propojení dvou mobilních telefonů je potřeba vyřešit problém určení typu zařízení, protože oba přístroje se mohou chovat jako USB host. Tento fakt ovšem nesmí na USB sběrnici nikdy nastat. Určení typu zařízení se děje hardwarově tím, že USB konektory obsahují další vodič nazvaný ID. Je-li v okamžiku připojení USB kabelu na tomto vodiči hodnota logické 1, zařízení se bude chovat jako USB device, hodnota logické 0 znamená, že zařízení se bude chovat jako USB host. Tyto hodnoty vstupu určuje připojený USB kabel. Režim OTG zavádí dva nové protokoly HNP (Host Negotiation Protocol) a SRP (Session Request

Protocol). Protokol HNP slouží pro účel záměny funkce zařízení. Budeme-li chtít propojit např. digitální fotoaparát s tiskárnou, tak po připojení USB kabelu by se mohla tiskárna nastavit jako USB host a fotoaparát jako USB device. Ovšem fotoaparát musí být nakonfigurován jako USB host, protože potřebuje řídit USB sběrnici. K tomuto účelu právě slouží HNP protokol, který zamění funkci obou zařízení tak, že fotoaparát se bude chovat jako USB host a tiskárna jako USB device. SRP protokol umožňuje vzdálené probuzení zařízení tím, že bude po vodiči VBUS vysílat impulsy, bude přerušovat napájení. Proto je nutné pro zařízení v OTG režimu zajistit obvod pro přerušování napájení.

1.2.2 Logická topologie

Logická topologie je pohled na fungování celé USB sběrnice, popisuje skutečnou komunikační strukturu sběrnice. Vše je znázorněno na následujícím obrázku:



Obrázek 5 Náhled logické struktury USB sběrnice

Schéma ukazuje nejjednodušší případ propojení zařízení typu USB host se zařízením USB device, kdy mezi těmito zařízeními není přítomen žádný USB HUB. Klientský software představuje určitou aplikaci, která chce používat USB sběrnici. Software USB systému si lze představit jako ovladač, který přijímá požadavky klientského softwaru, podle těchto požadavků dává příkazy pro rozhraní USB. USB rozhraní, ať již na straně USB host nebo na straně USB device, je tvořeno hardwarem zajišťujícím samotný přístup ke sběrnici. Logické rozhraní na straně USB device je

představováno jedinou částí, kterou je tzv. řídící koncový bod, který má obvykle adresu nula. Funkce koncového zařízení se skládá z celé řady dalších koncových bodů.

Koncový bod si lze představit jako část paměti zařízení, do které se může zapisovat nebo je možné z ní číst. Jediným koncovým bodem, kterým musí být vybaveno každé USB zařízení, je řídící koncový bod, ostatní koncové body již nemusí v zařízení být přítomny. Pomocí řídícího koncového bodu se provádí veškeré nastavení USB zařízení. Další koncové body jsou již závislé na dané funkci USB zařízení.

Vyžaduje-li klientský software vyslat nějaká data do části funkce koncového zařízení, provede to tak, že tato data zapíše do určitého koncového bodu, resp. na adresu koncového bodu. Z tohoto koncového bodu si data cílové zařízení vyzvedne a zpracuje. Naopak chce-li poslat nějaká data koncové zařízení (USB device), pouze je zapíše do svého koncového bodu, odkud si je poté USB host vyzvedne a zpracuje. Logickou topologií si tedy data předávají pouze určité části paměti obou zařízení. Tyto cesty pro přenos dat nazýváme roury, které ve skutečnosti fyzicky neexistují, ale představují jakési pomyslné spojení mezi klientským softwarem a funkcí koncového zařízení. Fyzicky data putují přes všechny části obou zařízení. Zvláštním případem roury je tzv. defaultní roura, která zprostředkovává základní komunikaci mezi USB zařízeními. Komunikace s řídícím koncovým bodem probíhá pomocí tzv. SETUP paketů.

1.3 Formy datových paketů

Před vysláním každého paketu se nejdříve vyšle bitová kombinace, která má za úkol synchronizovat hodinou frekvenci vysílače a přijímače. Protože sběrnice USB používá kódování NRZI, kde při výskytu logické nuly dochází ke změně stavu na datových vodičích, je prvotní bitová kombinace nastavena na 00000001_B. Při této kombinaci dojde 7x ke změně hodnoty na datových vodičích, v signálu se objeví celkem sedm hran signálu, na které se synchronizuje přijímač s frekvencí vysílače. Této bitové kombinaci se říká SYNC.

Aby se při přenosu dat udržela synchronizace, je vždy při výskytu šesti logických jedniček vyslána jedna logická nula. Tím se v signálu vyskytne změna úrovně na datových vodičích a přijímač má možnost zasynchronizovat svoji frekvenci.

Po vyslání kombinace SYNC následuje tzv. PID (Packet ID). PID má délku osm bitů. Z těchto osmi bitů nese užitečnou informaci pouze dolní čtveřice bitů, horní čtveřice bitů je pouze invertovaná dolní čtveřice bitů. PID obsahuje informaci o tom, jaký paket se bude přenášet. Následující výčet ukazuje nejběžnější případy označení PID:

Tabulka 1 Přehled základních typů PID

Název PIDu	Popis
OUT	Paket obsahuje adresu zařízení, adresu koncového bodu. Slouží pro přenos ze zařízení USB host do USB device
IN	Paket obsahuje adresu zařízení, adresu koncového bodu. Slouží pro přenos za zařízení USB device do USB host.
SOF	Obsahuje informaci o číslu rámce.
SETUP	Paket obsahuje adresu zařízení, adresu koncového bodu. Používá se výhradně při komunikaci s řídícím koncovým bodem.
DATA0	Následuje datový sudý paket.
DATA1	Následuje datový lichý paket.
ACK hadnshake	Nese pouze informaci o potvrzení příjmu datového paketu.
NAK handshake	Přijímací strana neakceptovala přijatá data.
STALL handshake	Koncový bod je zastaven nebo řídící koncový bod nemohl data zpracovat.

Pakety obsahující PID IN, OUT a SETUP slouží pro výběr daného zařízení a koncového bodu. Za těmito pakety mohou následovat ostatní typy PIDů (např. při přenosu dat pro koncový bod s adresou dva se nejdříve vybere dané zařízení a daný koncový bod, poté se pomocí PIDu IN nebo OUT přenesou potřebná data). V adrese koncového bodu se navíc přenáší informace o směru komunikace. Nastavení nejvyššího bitu adresy znamená, že komunikace bude probíhat směrem do zařízení USB host, z tohoto důvodu může být na USB sběrnici pouze 127 zařízení (adresa 0 je rezervovaná pro nově připojené, nezkonfigurované, zařízení).

Pro verzi USB 1.1 jsou všechny pakety vysílány v 1ms rámcích. Pro rychlost High speed je doba vysílání 125μs. Doby rámců nepředstavují dobu trvání rámce, ale četnost vysílání rámce.

Jediným paketem, který má přesně daný formát, je SETUP paket. Tento paket obsahuje vždy osm bajtů. Za tímto paketem mohou následovat další data, a to jak pro směr ze zařízení USB host, tak i z něj. SETUP paket má následující formát:

Tabulka 2 Formát SETUP paketu

Název položky	Velikost položky	Popis položky
bmRequestType	1 bajt	Určuje směr požadavku dat, komu je požadavek určen (koncovému bodu, zařízení, rozhraní) a typ požadavku.
bRequest	1 bajt	Udává přesný typ požadavku
wValue	2 bajty	Může obsahovat určitá malá data požadavku.
wIndex	2 bajty	Určuje číslo koncového bodu nebo rozhraní.
wLength	2 bajty	Délka dat následujících za SETUP paketem

Pomocí SETUP paketů se nejčastěji přenášejí tzv. USB funkce. Tyto funkce slouží pro nastavování parametrů USB zařízení nebo pro získávání informací o zařízení. Těmito funkcemi jsou např. SET_ADDRESS, GET_DESCRIPTOR.

Funkce SET_ADDRESS slouží pro nastavení adresy USB zařízení. Tato funkce má jednoduchý formát:

Tabulka 3 Ukázka USB funkce SET_ADDRESS

bmRequestType	bRequest	wValue	wIndex	wLength
00000000_B	SET_ADDRESS	Adresa zařízení	0	0

Položka bmRequestType určuje, že požadavek je určen pro USB zařízení, bRequest obsahuje kód funkce, proměnná wValue obsahuje novou adresu zařízení. Za tímto SETUP paketem již nebudou žádná data, proto je položka wLength nastavena na nulu.

Tabulka 4 Ukázka USB funkce GET_DESCRIPTOR

bmRequestType	bRequest	wValue	wIndex	wLength
10000000_B	GET_DESCRIPTOR	Typ a index deskriptoru	0	Délka deskriptoru

Parametr bmRequestType nyní ukazuje, že se vyžadují data pro zařízení USB host. Položka wValue obsahuje typ deskriptoru, např. deskriptor zařízení apod., ale také obsahuje index deskriptoru pro řetězcové deskriptory. Položka wLength určuje maximální velikost odesílaného deskriptoru.

Nejsou-li za SETUP paketem vyžadována žádná data, je nutné, aby zařízení USB device vyslalo tzv. paket nulové délky. Tento paket nenese žádná data.

Pakety s PID IN, OUT, SETUP, DATA0, DATA1 jsou zabezpečeny pomocí CRC (Cyclic Redundancy Check) kódu, který všechna přenášená data zabezpečuje. Velikost CRC kódu se pro jednotlivé typy PID liší.

1.4 Formy datových přenosů

Na sběrnici USB můžeme použít pouze čtyři typy přenosů dat. Jednotlivé typy se od sebe liší zabezpečením, možnou délkou a typem použití.

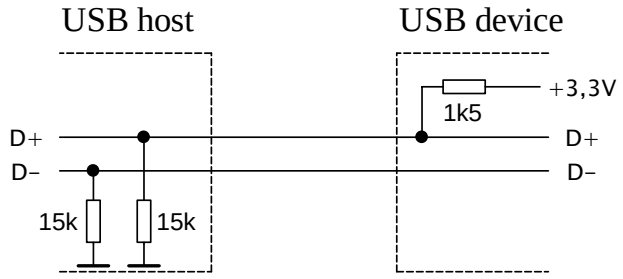
Hlavním druhem přenosu dat je řídící přenos, který se používá při komunikaci s řídícím koncovým bodem. Pro přenos velkých objemů dat slouží tzv. bulk přenos. U tohoto typu je zaručeno, že data budou doručena v původní podobě (např. Flash disky). Pro přenos krátkých rychlých zpráv je určen tzv. interrupt přenos, který používají např. klávesnice pro přenesení stisku klávesy. Posledním typem přenosu je tzv. izochronní přenos, který je určen pro přenos velkého objemu dat, u něhož záleží na rychlosti doručení a nemusí být zaručeno, že data dorazí v pořádku. Tohoto přenosu využívají výhradně mediální zařízení např. při přenosu obrazu nebo zvuku.

1.5 Rychlosti USB zařízení

Rychlosti USB zařízení jsou závislé na verzi USB sběrnice, kterou dané zařízení používá. Celkem existují již tři verze USB.

První verzí byla verze USB 1.1. Tato verze umožňuje dvě rychlosti. Rychlost Low speed s maximální přenosovou rychlostí 1,5Mb/s a rychlost Full speed s rychlostí až 12Mb/s. Druhou verzí je USB 2.0. Ta přidává k předchozí verzi další možnou rychlost zařízení, kterou je High speed s maximální rychlostí přenosu až 480Mb/s. Verze USB 1.1 a USB 2.0 jsou zpětně kompatibilní. V letošním roce přibyla poslední verze, kterou je USB 3.0. Ta výrazně zvyšuje přenosovou rychlost, a to až na 5Gb/s. S tím je také spojena změna kabeláže a princip přenosu bitů. Zatímco verze USB 1.1 a USB 2.0 používaly jen jeden diferenciální pár vodičů, verze USB 3.0 používá tento pár také pro zachování zpětné kompatibility, ale navíc používá další dva diferenciální páry. To má také za následek vznik nových typů konektorů. Bližší pohled na verzi USB 3.0 může poskytnout [5] nebo [6].

Nabízí se otázka, jak konkrétní rychlost, kterou je schopné zařízení USB device komunikovat, rozpoznat. U verze USB 1.1 se rychlost rozpoznává hardwarově již při připojení USB zařízení. Princip ukazuje tento obrázek:



Obrázek 6 Princip určení rychlosti USB zařízení

Na straně zařízení USB host jsou datové vodiče připojeny na zem pomocí dvojice rezistorů s hodnotou 15kΩ. Na straně USB device je nutné k jednomu datovému vodiči připojit rezistor s hodnotu 1,5kΩ. To, na který datový vodič se tento rezistor připojí, určuje rychlost zařízení. Připojení rezistoru 1,5kΩ na vodič D+ vybere rychlost Full speed, připojení na vodič D– určí rychlost Low speed. Jelikož verze USB 2.0 zachovává zpětnou kompatibilitu s verzí USB 1.1, tento samý princip platí i pro tuto verzi. Navíc verze USB 2.0 zavádí nový protokol, který určí, jestli je zařízení schopno pracovat na rychlosti High speed. Pokud se připojí zařízení verze USB 1.1 ke sběrnici USB 2.0, toto zařízení nebude danému protokolu rozumět a sběrnice se automaticky přepne do verze USB 1.1.

1.6 Třídy USB zařízení

Tvůrci USB zařízení navrhly řadu tříd zařízení. Každá třída může obsahovat řadu podtříd, ty jsou dále rozděleny podle protokolů. Třídu USB zařízení lze chápat jako soupis pravidel, které určují, jak se má zařízení chovat, které koncové body má mít implementovány, typ koncových bodů, definuje vlastní USB funkce atd. Pravidla USB třídy popisují i formát rámců dat přenášených prostřednictvím koncových bodů.

Tabulka 5 Přehled základních USB tříd

Číslo USB třídy	Deskriptor	Popis
00_H	Zařízení	Základní třída, informaci o třídě nesou deskriptory rozhraní
01_H	Rozhraní	Audio třída
02_H	Zařízení, rozhraní	Zařízení určená pro přenos dat, také pro CDC (Comunnication Devices Class)
03_H	Rozhraní	HID (Human Interface Device)
07_H	Rozhraní	Tiskárny
08_H	Rozhraní	Např. Flash disky (Mass Storage)
09_H	Zařízení	USB HUB
$0A_H$	Rozhraní	Pro rozhraní přenášející data u CDC zařízení.
$0E_H$	Rozhraní	Pro video zařízení
FF_H	Zařízení, rozhraní	Zvláštní případ třídy, kdy o funkčnosti zařízení rozhoduje VID a PID zařízení.

Tabulka ukazuje výpis části USB tříd. První sloupec udává kód dané třídy, druhý sloupec obsahuje informaci o tom, ve kterém deskriptoru se musí číslo USB třídy nacházet. Poslední sloupec obsahuje popis dané třídy.

1.7 Deskriptory zařízení

Deskriptor je prostředek, jenž poskytuje informace o USB zařízení a jeho funkci. Tyto informace vyžaduje zařízení USB host v době připojení zařízení. Informace jsou nutné proto, aby bylo jasné, o jaké zařízení se jedná, do jaké třídy patří apod.

USB host také musí vědět velikosti jednotlivých koncových bodů. Pro všechny tyto účely je vytvořena řada deskriptorů.

1.7.1 Deskriptor zařízení

Deskriptor zařízení má následující strukturu:

```
byte bLength // Počet bajtů deskriptoru zařízení
byte bDescriptorType // Typ deskriptoru = 1
uint bcdUSB // Verzi USB sběrnice pro USB 2.0 = 20H
byte bDeviceClass // Třída USB zařízení
byte bDeviceSubClass // Podtřída USB zařízení
byte bDeviceProtocol // Protokol zařízení
byte bMaxPacketSize0 // Velikost koncového bodu nula
uint idVendor // Identifikátor výrobce VID (Vendor ID)
uint idProduct // Identifikátor produktu PID (Product ID)
uint bcdDevice // Verze zařízení – hexadecimální hodnota
byte iManufacturer // Index řetězcového deskriptoru jména výrobce
byte iProduct // Index řetězcového deskriptoru názvu produktu
byte iSerialNumber // Index řetězcového deskriptoru sériového čísla
byte bNumConfigurations
```

Deskriptor zařízení nese všechny nejnutnější informace o USB zařízení. Po příjmu tohoto deskriptoru již bude USB host předběžně vědět, s jakým zařízením komunikuje.

1.7.2 Konfigurační deskriptor

Tento deskriptor je druhým nejdůležitějším a také jediným, který obsahuje další typy deskriptorů. Nese informace o velikosti odběru proudu zařízení, o typu napájení a o celkovém počtu rozhraní.

```
byte bLength // Délka samotného konfiguračního deskriptoru
byte bDescriptorType // Typ deskriptoru = 2
uint wTotalLength // Celková délka deskriptoru se všemi ostatními deskriptory
byte bNumInterfaces // Počet rozhraní v konfiguračním deskriptoru
byte bConfigurationValue // Číslo předané USB funkci SET_CONFIGURATION
byte iConfiguration // Index pro řetězcový popis deskriptoru
byte bmAttributes // Parametry napájení, zařízení je napájeno z USB sběrnice apod.
byte bMaxPower // Maximální proudový odběr zařízení
```

1.7.3 Deskriptor rozhraní

Rozhraní slučuje určitý počet koncových bodů jedné funkce. Např. bude-li jedno zařízení představovat USB klávesnici a USB myš, tak zařízení bude obsahovat dvě rozhraní a každé z těchto rozhraní bude mít např. dva koncové body. Pomocí tohoto deskriptoru bude USB host vědět, se kterými koncovými body má komuniko-

vat, aby zajistil správnou funkci obou zařízení. Tento deskriptor je součástí konfiguračního deskriptoru.

```
byte bLength // Délka deskriptoru rozhraní
byte bDescriptorType // Typ deskriptoru = 4
byte bInterfaceNumber // Číslo rozhraní, používáno jako parametr USB funkcí
byte bAlternateSetting // Položka pro možné nastavení rozhraní
byte bNumEndpoints // Počet koncových bodů patřících k tomuto rozhraní
byte bInterfaceClass // USB třída rozhraní
byte bInterfaceSubClass // USB podtřída pro rozhraní
byte bInterfaceProtocol // Protokol pro rozhraní
byte iInterface // Index pro popis rozhraní
```

1.7.4 Deskriptor koncových bodů

Deskriptor koncových bodů udává velikosti jednotlivých koncových bodů, jejich typ, směr komunikace a adresu koncového bodu. Tento deskriptor musí následovat hned po deskriptoru rozhraní, ke kterému koncové body patří. Deskriptor koncových bodů je tedy součástí konfiguračního deskriptoru.

```
byte bLength // Délka deskriptoru koncových bodů
byte bDescriptorType // Typ deskriptoru = 5
byte bEndpointAddress // Adresa a směr koncového bodu
byte bmAttributes // Typ koncového bodu – řídící, bulk, ...
uint wMaxPacketSize // Velikost koncového bodu
byte bInterval // Doba čekání na data koncového bodu
```

1.7.5 Řetězcové deskriptory

Tyto deskriptory obsahují textový řetězec, který slouží k popisu dané části zařízení. Řetězcovým deskriptorem může být např. popsáno jméno výrobce, název produktu nebo také samotné rozhraní apod. Formát těchto deskriptorů je následující:

```
byte bLength // Celková délka deskriptoru
byte bDescriptorType // Typ deskriptoru = 3
byte char0 // První znak
...
byte charN // Poslední znak
```

Po odeslání všech druhů deskriptorů má už zařízení USB host všechny informace o připojeném zařízení. Tím je ukončen tzv. proces identifikace zařízení, nebo je také možné hovořit o tom, že je zařízení správně zkonfigurováno.

2 Dostupné obvody pro USB sběrnici

Na trhu je k dostání řada obvodů, které určitým způsobem zprostředkovávají komunikaci s USB sběrnicí. Nejčastěji se lze setkat s obvody ve formě různých převodníků. Jako příklad můžeme uvést převodník z USB sběrnice na UART (Universal Asynchronous serial Reciever and Transmitter). Mezi ty společnosti, které nabízejí dané převodníky, patří společnost FTDI (Future Technology Devices International). K dostání jsou i obvody, které umožňují libovolnou funkčnost, např. mikrokontroléry s implementovaným USB řadičem, ovšem tyto obvody vyžadují značnou znalost celé problematiky, protože je nutné vytvořit nový obslužný firmware. Mikrokontroléry s USB řadičem nabízí ve svém sortimentu např. společnost Microchip nebo Atmel.

2.1 Převodníky společnosti FTDI

Tato společnost již řadu let nabízí širokou škálu převodníků, které zprostředkovávají komunikaci po USB sběrnici. Celý přehled výrobků včetně popisu lze nalézt v [7]. Tyto obvody slouží nejčastěji jako převodníky z USB sběrnice na UART nebo na tzv. paralelní port. Převodníky se chovají jako obecné USB zařízení, které přijatá data přemění na jiný typ přenosu. Obvody sloužící jako sériový port mají vyvedeny všechny vodiče jako skutečný sériový port. Obvody paralelního portu se nechovají jako klasický paralelní port, ale chovají se jako paměť typu FIFO (First In First Out). Všechny obvody umožňují nastavení některých položek deskriptorů pomocí příkazů, které zapisují data do paměti EEPROM.

Společnost FTDI nabízí ke svým obvodům volně dostupné ovladače, které zprostředkovávají vlastní komunikaci se zařízením. Ovladače fungují ve všech běžných operačních systémech Microsoft Windows XP, Microsoft Windows Vista, Linux, apod. Existují dva typy ovladačů. Jeden zprostředkovává přímý přístup k USB zařízení a druhý se chová jako virtuální sériový port. Při použití přímého přístupu k USB zařízení lze použít zvláštní režim tzv. BitBang mód. V tomto módu se odpojí vnitřní převodník a vývody obvodu se dají naprogramovat jako libovolný vstup a výstup.

Společnost FTDI nabízí několik řad převodníků. V každé řadě se většinou vyskytují dva typy obvodů. Jedním je převodník na UART, čili sériový port, druhým typem převodníku je tzv. paralelní port.

2.1.1 Řada B

Tato řada je jednou z nejstarších a obsahuje dva základní typy obvodů, obvod pro sériový port FT232BM/BL/BQ (liší se pouze v provedení pouzdra obvodu) a obvod pro paralelní port FT245BM/BL/BQ. Obvody FT232Bx mohou komunikovat i po sběrnici RS485.

Vlastnosti řady B:

- Napájení +5V.
- Vnitřní regulátor napětí +3,3V, zatížitelnost 50mA.
- Potřeba externí paměti EEPROM.
- Možnost BitBang módu.
- Maximální přenosová rychlost pro sériový port 3Mb/s, pro paralelní port 1Mb/s.
- Výstupy pro LED diody indikující přenos dat.

2.1.2 Řada R

Tato řada je nástupcem řady B. Také obsahuje dva druhy zařízení. Pro sériový port je to obvod FT232R a pro paralelní port je to obvod FT245R. Všechny obvody mají řadu vylepšení oproti předcházející řadě. Např. obvody mají integrovaný vlastní oscilátor, i paměť EEPROM je integrována dovnitř obvodu, tím se výrazně zjednodušuje návrh plošného spoje. BitBang režim lze použít na výstupy samotného převodníku nebo na jiné vývody, které se dají různě konfigurovat. Umožňují např. vyvedení frekvence oscilátoru nebo nastavit chování výstupů v závislosti na stavu obvodu např. signalizace příjmu nebo vysílání dat.

Vlastnosti řady R:

- Napájení +5V.
- Vnitřní regulátor napětí +3,3V.
- Vnitřní paměť EEPROM.

- Rozšířený režim BitBang.
- Maximální přenosová rychlost pro sériový port 3Mb/s, pro paralelní port 1Mb/s.

2.1.3 Řada FT2232

Tato řada se od předchozích velice liší. Implementuje celkem dva převodníky. Každý z těchto převodníků lze nezávisle nastavit jako sériový nebo paralelní port. Nejzajímavější novinkou této řady je nové rozhraní MPSSE (Multi-Protocol Synchronous Serial Engine Interface), které umožňuje implementaci různých synchronních rozhraní jako je např. JTAG, I2C nebo SPI.

Paměť EEPROM a oscilátor nejsou umístěny uvnitř obvodu, ale musí být připojeny externě.

2.1.4 Řady FT2232H a FT4232H

Tyto řady jsou poslední v nabídce společnosti FTDI. Do jisté míry navazují na řadu FT2232. Největší rozdíl ale je v tom, že tyto obvody nové řady umožňují rychlost zařízení High speed, tedy až 480Mb/s. Řada FT2232H je stejná s řadou FT2232, jen rychlost zařízení je vyšší. Řada FT4232H obsahuje již čtyři samostatná zařízení.

Dalším již méně významným rozdílem oproti předcházejícím řadám je ten, že obvody již nejsou napájeny +5V, ale pouze +3,3V. Pro napájení jádra obvodů je potřeba napětí +1,8V, ale obvody obsahují regulátor napětí z +3,3V na +1,8V. Paměť EEPROM i oscilátor musí být připojeny externě.

2.1.5 Řada Vinculum

Řada Vinculum má jediného zástupce. Tím je obvod VNC1L s vyvedenými dvěma USB porty. Tato řada se od ostatních opět liší. Zatímco předchozí řady fungovaly pouze jako USB device, obvod VNC1L se chová jako zařízení USB Host. Popis této řady lze najít v [8].

Pro obvod VNC1L nabízí společnost FTDI řadu volně stažitelných firmwarů, včetně programů umožňujících nahrání nového firmwaru. Volně ke stažení je i pro-

gram pro drobné úpravy firmwaru. Mezi úpravy např. patří volba přenosové rychlosti, řízení toku dat, atd. Každý firmware se liší v použitelnosti obvodu. Následující přehled shrnuje všechny druhy firmwaru:

Tabulka 6 Přehled firmwaru pro obvod VCN1L

Název firmwaru	Popis
VDAP	Umožňuje připojit širokou škálu USB zařízení. Např. zařízení typu HID, HUB, CDC, … Na jeden port lze připojit Flash disk.
VDIF	K jednomu portu lze připojit Flash disk a ke druhému zařízení společnosti FTDI.
VMSC	Umožňuje připojit MP3 přehrávač.
VDPS	Lze k jednomu portu připojit Flash disk. Druhý port slouží jako USB zařízení.
VCDC	Lze připojit pouze zařízení třídy CDC.
VDFC	Umožňuje kopírování z jednoho Flash disku na druhý.

Obvod VCN1L lze ovládat pomocí sériového rozhraní (UART), pomocí paralelního rozhraní (FIFO) nebo pomocí sériového rozhraní SPI. Volbu používaného rozhraní je možné zvolit hardwarově. Samotný obvod lze řídit sadou příkazů, které se pro různé typy firmwaru liší. Pomocí této sady příkazů lze např. u firmwaru VDAP používat libovolné USB zařízení, protože existuje příkaz pro poslání samotného SETUP paketu. Zařízení lze tedy ovládat na té nejnižší úrovni.

2.2 Mikrokontroléry s USB řadičem společnosti ATMEL

Firma Atmel nabízí ve svém sortimentu řadu mikrokontrolérů s jádrem AVR, které mají přímo na svém čipu implementovaný USB řadič.

Tyto mikrokontroléry a jejich základní vlastnosti jsou uvedeny v následujícím přehledu:

Tabulka 7 Přehled mikrokontrolérů s USB řadičem firmy Atmel

Název mikro-kontroléru	Velikosti pamětí		Napájecí napětí	Maximální frekvence	Funkce USB
AT90USB82	FLASH	8kB	2,7 – 5,5V	16MHz	Device
	SRAM	512B			
	EEPROM	512B			
AT90USB162	FLASH	16kB	2,7 – 5,5V	16MHz	Device
	SRAM	512B			
	EEPROM	512B			
AT90USB646	FLASH	64kB	2,7 – 5,5V	16MHz	Device
	SRAM	4kB			
	EEPROM	2kB			
AT90USB647	FLASH	64kB	2,7 – 5,5V	16MHz	Device, Host
	SRAM	4kB			
	EERPOM	2kB			
AT90USB1286	FLASH	128kB	2,7 – 5,5V	16MHz	Device
	SRAM	8kB			
	EERPOM	4kB			
AT90USB1287	FLASH	128kB	2,7 – 5,5V	16MHz	Device, Host
	SRAM	8kB			
	EERPOM	4kB			
ATmega8U2	FLASH	8kB	2,7 – 5,5V	16MHz	Device
	SRAM	512B			
	EERPOM	512B			
ATmega16U2	FLASH	16kB	2,7 – 5,5V	16MHz	Device
	SRAM	512B			
	EERPOM	512B			
ATmega16U4	FLASH	16kB	2,7 – 5,5V	16MHz	Device
	SRAM	2,5kB			
	EERPOM	1kB			
ATmega32U2	FLASH	32kB	2,7 – 5,5V	16MHz	Device
	SRAM	1kB			
	EERPOM	1kB			
ATmega32U4	FLASH	32kB	2,7 – 5,5V	16MHz	Device
	SRAM	2,5kB			
	EERPOM	1kB			

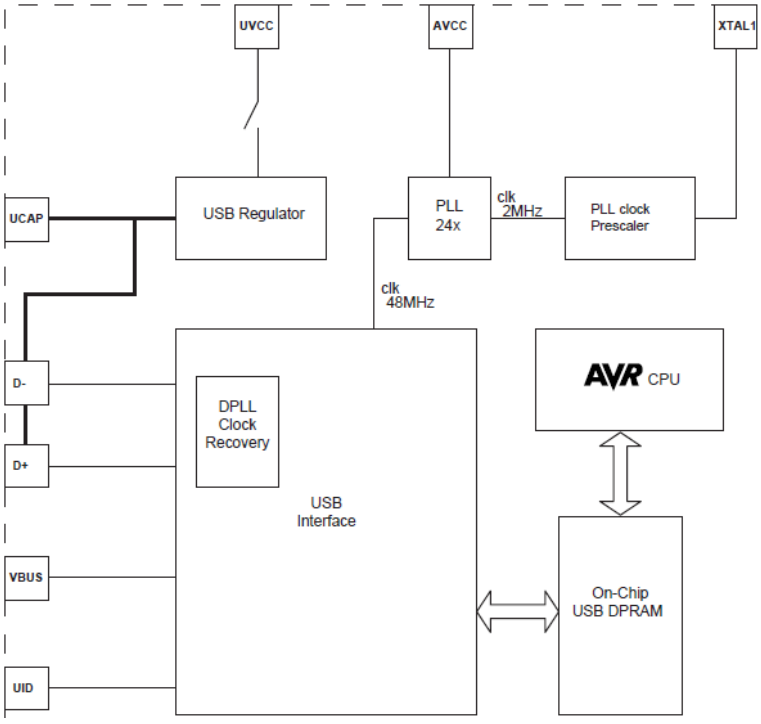
Všechny výše jmenované mikrokontroléry jsou založeny na osmibitovém jádru AVR a obsahují všechny standardní periferie jako je např. jednotka USART, SPI, časovače, vstupně/výstupní porty, apod.

3 Mikrokontrolér AT90USB647

Jak již bylo dříve zmíněno, jedná se o klasický osmibitový mikrokontrolér založený na jádru AVR firmy ATMEL. Má všechny běžné periferie, které jsou pro tuto rodinu mikrokontrolérů běžné. Navíc je vybaven hardwarovým USB řadičem, který se tak stává další periferií. Tento USB řadič umožňuje vytvářet zařízení USB device, ale také i zařízení typu USB host, který podporuje režim OTG. Jako každá periferie se ovládá pomocí určité sady registrů a systémem přerušení.

Problém řízení USB řadiče lze rozdělit na tři samostatné kategorie podle toho, jakou část chceme ovládat. Jedná se tedy o ovládání hlavních částí USB řadiče, jako je např. monitoring napětí na USB sběrnici, zapínání vnitřního regulátoru napětí apod. Další částí je ovládání zařízení USB device. Poslední částí je řízení zařízení USB host. Celý popis lze najít v [9].

3.1 Blokové schéma USB řadiče



Obrázek 7 Blokové schéma USB řadiče [9]

Popis vývodů USB řadiče:

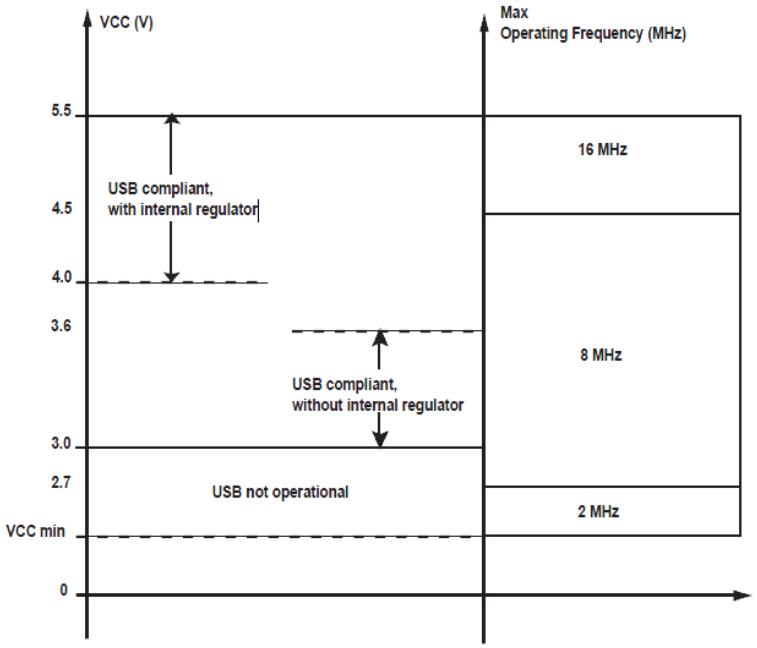
- **UVCC** – napájení pro interní regulátor napětí
- **XTAL1** – zdroj hodinové frekvence
- **UCAP** – slouží pro připojení externího filtračního kondenzátoru, který filtruje napětí vytvořené interním regulátorem napětí
- **D-** a **D+** - datové vodiče USB sběrnice

Popis jednotlivých dílčích bloků:

- **USB Interface** – blok, který se stará o veškerou USB komunikaci.
- **PLL Clock Prescaler** – z přivedené frekvence na vývod XTAL1 vytváří frekvenci 2MHz. Tento blok se dá samostatně nastavit pomocí svého registru viz. níže.
- **PLL 24x** – násobí frekvenci vytvořenou v bloku PLL Clock Prescaler dvacetičtyřmi na výslednou frekvenci 48MHz±0,25% pro USB Interface. Tento blok se u jednotlivých typů mikrokontrolérů může lišit. Jeho funkce je ve všech případech stejná jen velikost násobení frekvence se liší.
- **DPLL** – blok pro řízení frekvence podle USB specifikace.
- **Regulator** – přeměňuje napětí přivedené na UVCC (+5V) na napětí +3,3V, které je potřebné pro správnou funkci USB řadiče.
- **On-Chip USB DPRAM** – zvláštní paměť RAM pro účely USB řadiče. V této paměti jsou uloženy data koncových bodů nebo data paměťových rour.

3.2 Připojení mikrokontroléru k USB sběrnici

Připojení mikrokontroléru k USB sběrnici je závislé na způsobu napájení, na faktu, zda je mikrokontrolér napájen z externího zdroje nebo z USB sběrnice, a na velikosti napájecího napětí. Na velikosti napájecího napětí je také závislá hodinová frekvence mikrokontroléru. Všechny tyto údaje jsou shrnuty v následujícím obrázku:



Obrázek 8 Závislost napájecího napětí na frekvenci mikrokontroléru [9]

Z něho vyplývá, že pokud je velikost napájecího napětí do +3V, USB řadič je vyřazen z činnosti. Naopak je-li napájecí napětí v rozsahu +3V až +3,6V může USB řadič vykonávat svoji činnost na frekvenci 8MHz, v tomto rozsahu napětí musí být mimo činnost vnitřní regulátor napětí. Poslední napěťový rozsah, ve kterém může USB řadič pracovat, je +4V až +5,5V. V tomto rozsahu musí být naopak vnitřní regulátor napětí zapnut. Frekvence 16MHz může být přivedena na mikrokontrolér pouze při napájecím rozsahu +4,5V až +5V.

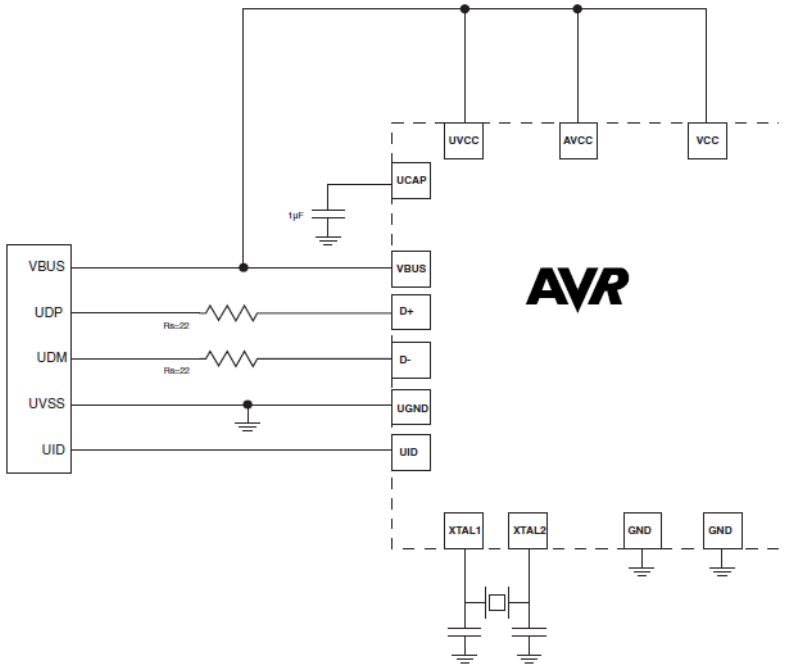
Pro nejčastěji v praxi používané napájecí rozsahy tedy platí:

Tabulka 8 Souhrn nastavení pro používané napájecí rozsahy

Napájecí napětí	Maximální frekvence mikrokontroléru	Funkce vnitřního regulátoru napětí
+3,3V	8MHz	Vypnuto
+5V	16MHz	Zapnuto

3.2.1 Zapojení mikrokontroléru v závislosti na napájecím napětí

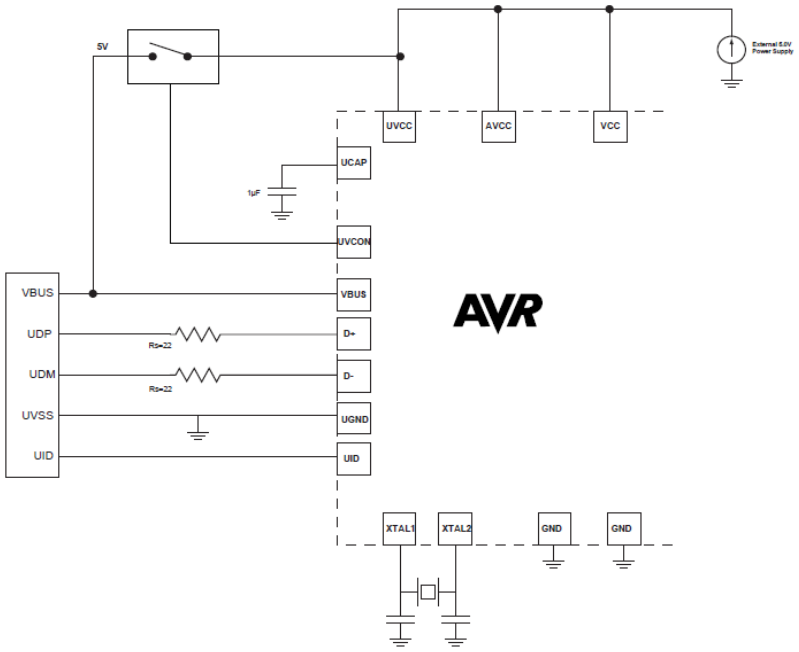
Následující obrázky ukazují způsob zapojení napájení mikrokontroléru, kdy plní funkci USB device.



Obrázek 9 Připojení mikrokontréru s napájením z USB sběrnice [9]

Tento obrázek ukazuje připojení mikrokontroléru k USB sběrnici v případě, že je napájen přímo z USB sběrnice. Na vývod UVCC je přivedeno napětí o velikosti +5V z USB sběrnice, které se pomocí vnitřního regulátoru zmenší na napětí +3,3V, a to je vyvedeno na vývod UCAP, kde je připojen filtrační kondenzátor.

Další zapojení je pro případ, kdy je mikrokontrolér v režimu USB host:



Obrázek 10 Zapojení mikrokontroléru v režimu USB host [9]

V tomto zapojení musí být mikrokontrolér napájen z externího zdroje napětí. Protože obvod musí také být schopen napájet připojené USB zařízení, je přidán obvod „spínače“, který je ovládán vývodem UVCON. Toto řízení je v obvodu zařazeno z toho důvodu, že mikrokontrolér v režimu USB host může také fungovat jako zařízení podporující OTG režim. Režim OTG přímo vyžaduje řízení napájecího napětí.

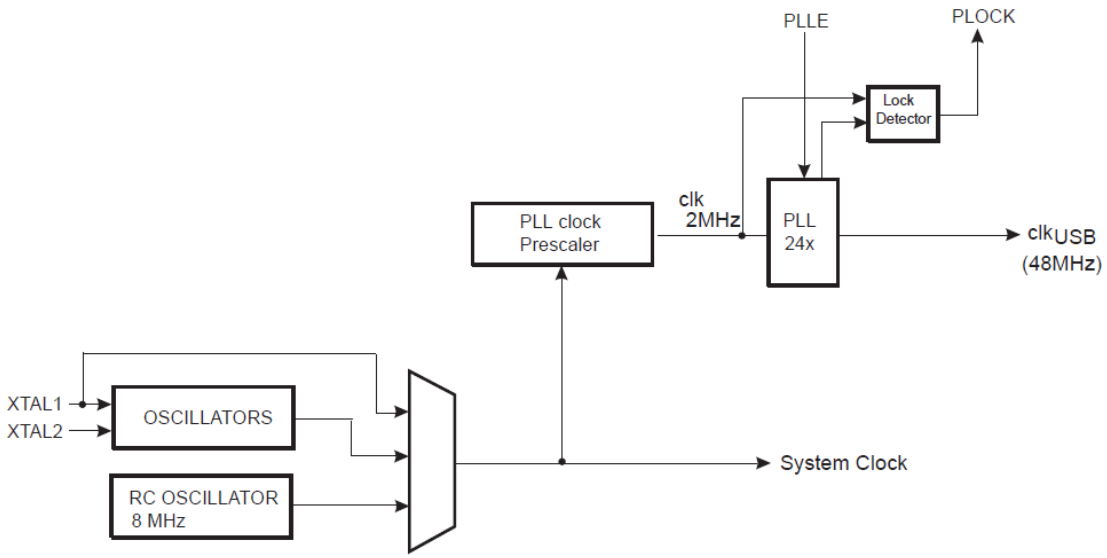
Pro režim OTG je také navíc nutné připojit vývod UID ke konektoru USB. Význam tohoto vývodu byl popsán v kapitole 1.2.1.

Výrobce mikrokontroléru také doporučuje správnou velikost některých součástek:

- Velikost sériových rezistorů Rs je 22Ω ±5%.
- Doporučená velikost kondenzátoru přivedeného na vývod UCAP je 1μF ±10%.
- Velikost kondenzátoru připojeného mezi VBUS a GND je 10μF.

3.3 Jednotka PLL

Hodinová frekvence mikrokontroléru je ve většině případů několikrát menší než potřebná frekvence pro USB řadič. Z tohoto důvodu má mikrokontrolér implementovanou jednotku PLL, která hodinovou frekvenci mikrokontroléru vynásobí a vytvoří tak potřebnou frekvenci. Blokové schéma této jednotky ukazuje následující obrázek:



Obrázek 11 Blokové schéma jednotky PLL [9]

Z něj vyplývá, že hodinová frekvence (System Clock) je přivedena na děličku frekvence. Na výstupu této děličky musí být frekvence 2MHz, která je přivedena na násobičku frekvence. Tato násobička má pevně nastavenou hodnotu násobení na 24x. Na výstupu násobičky je nutná frekvence 48MHz. Z těchto důvodů může mít hodinová frekvence mikrokontroléru pouze určité hodnoty, které jsou 8MHz a 16MHz. Obvod PLL navíc obsahuje možnost kontroly, jestli je frekvence na výstupu již ustálena. Pro účely nastavování a kontroly slouží registr PLLCSR:

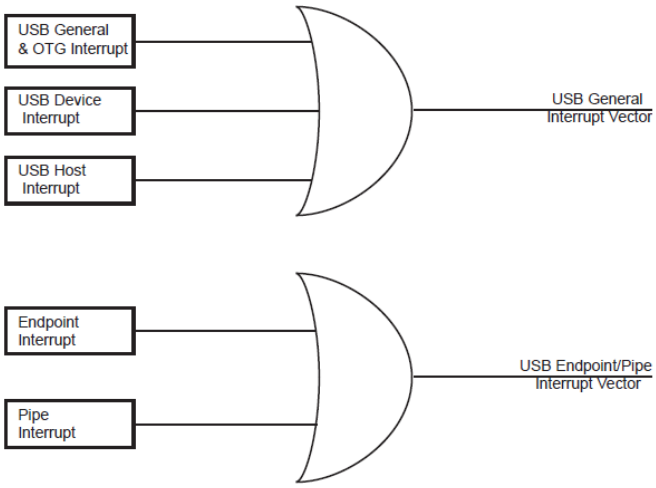
Bit	7	6	5	4	3	2	1	0	
$29 ($29)				PLLP2	PLLP1	PLLP0	PLLE	PLOCK	PLLCSR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0/1	0	

Obrázek 12 Ukázka registru PLLCSR [9]

- PLLP2:PLLP0 – nastavují hodnotu děličky kmitočtu.
- PLLE – povoluje celou jednotku PLL.
- PLOCK – kontroluje ustálení frekvence. Hodnota 1 znamená, že frekvence je stabilní. Pozn.: doba ustálení může trvat kolem 100ms.

3.4 Systém přerušení

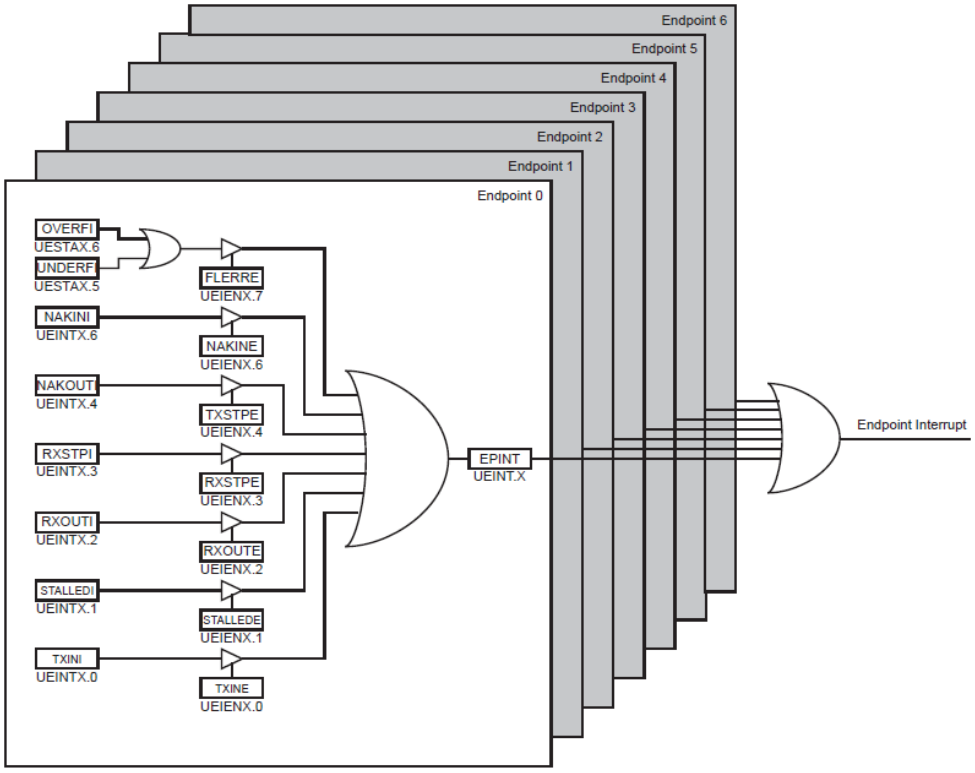
Pro obsluhu přerušení je mikrokontrolér vybaven celkem dvěma vektory přerušení, jak naznačuje následující obrázek:



Obrázek 13 Rozdělení vektorů přerušení USB řadiče [9]

Jeden vektor přerušení (USB General Interrupt Vector) obsluhuje události vzniklé v souvislosti s USB sběrnicí, jako je např. VBUSTI – změna připojení napětí na USB sběrnici, IDTI – hardwarová změna typu (Host, Device) připojeného zařízení, … Ale také zpracovává události USB sběrnice týkající se samotného USB device. Mezi tato přerušení patří např. SOFI – detekce Start Of Frame na sběrnici, WAKEUPI – probuzení USB sběrnice, … Tento vektor přerušení zpracovává řadu přerušení, proto je nutné v rutině přerušení zkontrolovat, ke kterému přerušení opravdu došlu a zpracovat jej.

Druhý vektor přerušení (USB Endpoint/Pipe Interrupt Vector) zpracovává události spojené s koncovými body v případě USB device nebo také s rourami v případě konfigurace jako USB host. Následující obrázek znázorňuje události obsluhované tímto vektorem přerušení pro zařízení USB device:



Obrázek 14 Struktura přerušení pro koncové body [9]

Z obrázku je vidět, že pro všechny koncové body existuje pouze jeden vektor přerušení. I pro tento vektor přerušení platí, že zpracovává několik přerušení, je tedy nutné rozpoznat, které přerušení nastalo, ale navíc musíme zjistit, kterého koncového bodu se týká. Pro tento případ existuje registr UEINT:

Bit	7	6	5	4	3	2	1	0	
	-	EPINT D6	EPINT D5	EPINT D4	EPINT D3	EPINT D2	EPINT D1	EPINT D0	UEINT
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

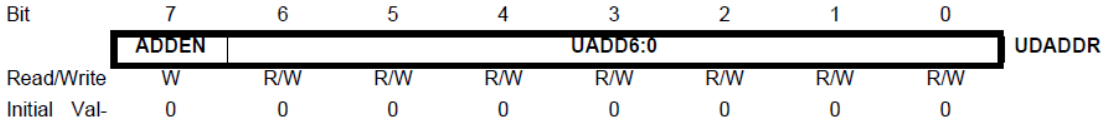
Obrázek 15 Ukázka registru UEINT [9]

Jednotlivé bity tohoto registru představují koncové body. Pokud vyvolal přerušení např. koncový bod 0, bude nastaven bit EPINT D0. Pro koncový bod 2 bude nastaven bit EPINT D2 atd.

Význam jednotlivých bitů, které vyvolávají a povolují přerušení, je popsán v podkapitolách zabývajících se problematikou, kterou tyto bity řeší.

3.5 Nastavení adresy USB zařízení

Proces nastavení adresy USB zařízení nastává těsně po připojení USB zařízení k zařízení typu USB host. Novou adresu zařízení obdrží v SETUP paketu. Pro nastavení adresy slouží následující registr:



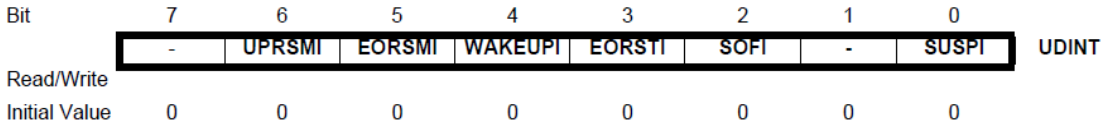
Obrázek 16 Ukázka registru UDADDR [9]

- ADDEN – nastavením tohoto bitu se aktivuje nová adresa.
- UADD6:0 – obsahují adresu zařízení.

Po resetu má zařízení adresu 0. Přijde-li SETUP paket s novou adresou, uloží se adresa do registru UDADDR, ale bit ADDEN se ponechá v hodnotě nula. Poté zařízení vyšle paket nulové délky pro potvrzení přijatého SETUP paketu. Nyní je již možné nastavit bit ADDEN pro povolení nové adresy, tím je proces nastavení nové adresy ukončen.

3.6 Kontrola stavu USB sběrnice

USB řadič v mikrokontroléru AT90USB647 obsahuje obvody pro kontrolu stavu USB sběrnice a vyhodnocení těchto stavů zprostředkovává programátorům prostřednictvím následujících registrů:



Obrázek 17 Ukázka registru UDINT [9]

- UPRSMI a EORSMI – souvisí se vzdáleným probuzením zařízení USB host z režimu SUSPEND.
- WAKEUPI – nastaven hardwarově v případě, že USB řadič detekuje stav WAKEUP – opětovné zprovoznění sběrnice. Nutné nulovat softwarově.
- EORSTI – nastaven hardwarově, kdy se detekuje reset na USB sběrnici. Nutné nulovat softwarově.

- SOFI – nastaven, když je detekován PID SOF.
- SUSPI – nastaven v okamžiku detekce uspání USB sběrnice, tj. v okamžiku, kdy je sběrnice v klidu po dobu tří period rámce, čili po dobu 3ms.

Všechny bity výše popsaného registru mohou vyvolat přerušení. Jednotlivá přerušení se povolují v registru UDIEN:

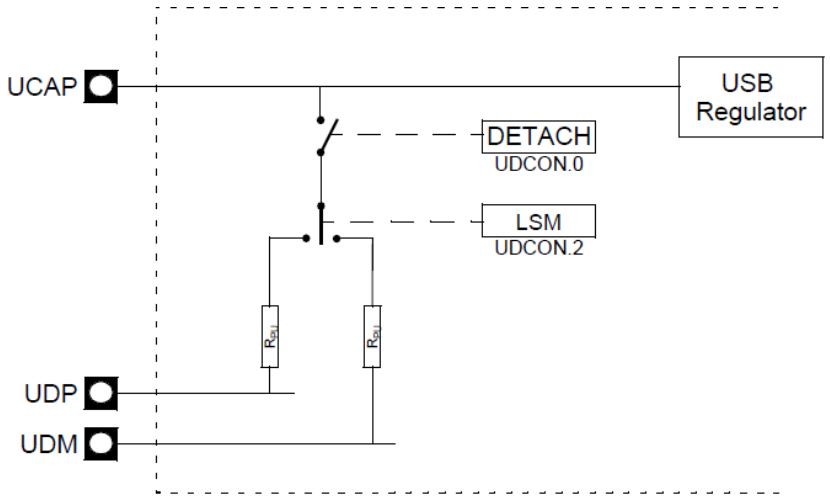
Bit	7	6	5	4	3	2	1	0	
	-	UPRSME	EORSME	WAKEUPE	EORSTE	SOFE	-	SUSPE	UDIEN
Read/Write									
Initial Value	0	0	0	0	0	0	0	0	

Obrázek 18 Ukázka registru UDIEN [9]

- Bity přesně kopírují bity registry UDINT.

3.7 Výběr rychlosti zařízení

Tento typ mikrokontroléru umožňuje pouze dvě rychlosti zařízení, tj. zařízení typu Low speed a Full speed. V kapitole 1.5 se uvádí, že USB host rozeznává typy zařízení podle pull-up rezistoru připojeného na jeden z datových vodičů. Tento mikrokontrolér tedy musí daný princip podporovat. Způsob vnitřního zapojení je znázorněn na následujícím obrázku:



Obrázek 19 Vnitřní způsob zapojení volby rychlosti USB zařízení [9]

Z něj je zřejmé, že bitem LSM se vybírá, který rezistor se připojí na napájecí napětí, a zároveň se tím vybere i rychlost zařízení. Bitem DETACH se vybraný rezistor připojí na napájecí napětí. Tyto bity lze nalézt v registru UDCON:

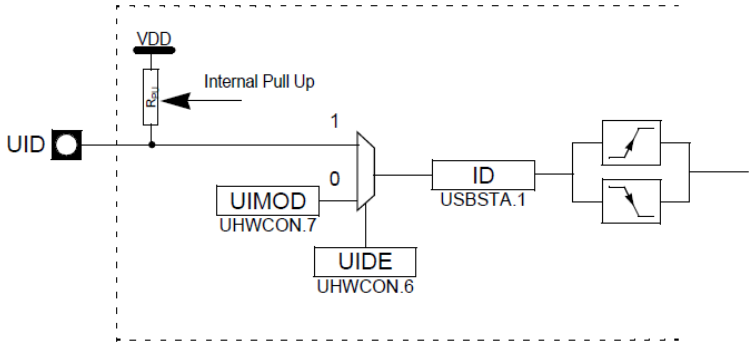
Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	LSM	RMWKUP	DETACH	UDCON
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	1	

Obrázek 20 Ukázka registru UDCON [9]

- LSM – = 1 pro Low speed, = 0 pro Full speed zařízení.
- RMWKUP – umožňuje vzdálené probuzení zařízení USB host v režimu SUSPEND.
- DETACH – vynulováním připojí vnitřní rezistory k napájecímu napětí.

3.8 Výběr typu zařízení

V kapitole 3.1 bylo zmíněno, že tento mikrokontrolér umožňuje jak funkci USB device, tak i funkci zařízení v režimu USB host s možností volby OTG režimu. V kapitole 1.2.1 byla funkce režimu OTG vysvětlena. Pro případ, že je nutné naprogramovat mikrokontrolér pouze jako USB device, existuje v mikrokontroléru hardwarová podpora vyřazení funkce vývodu UID. Vnitřní zapojení obvodu pro výběr typu zařízení je na následujícím obrázku:



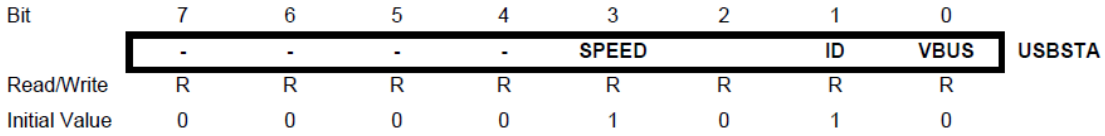
Obrázek 21 Vnitřní zapojení pro identifikaci typu USB zařízení [9]

Funkci vývodu UID lze zcela vyřadit pomocí bitu UIDE. Bude-li bit UIDE nastaven na hodnotu 0, bude typ zařízení vybrán podle hodnoty bitu UIMOD. Bude-li bit UIDE nastaven na 1, bude typ zařízení vybrán podle hodnoty vývodu UID. Bit ID zobrazuje výslednou hodnotu typu zařízení. Tyto bity lze nalézt v následujících registrech:

Bit	7	6	5	4	3	2	1	0	
	UIMOD	UIDE		UVCONE				UVREGE	UHWCON
Read/Write	R/W	R/W	R	R/W	R	R	R	R/W	
Initial Value	1	0	0	0	0	0	0	0	

Obrázek 22 Ukázka registru UHWCON [9]

- UIMOD – = 1 pro USB zařízení, = 0 pro USB Host.
- UIDE – = 1 typ zařízení vybírá vývod UID, = 0 typ zařízení vybírá bit UIMOD.
- UVCONE – povoluje funkci vývodu UVCON. Pouze v režimu USB Host.
- UVREGE – povoluje nebo zakazuje funkci vnitřního regulátoru napětí.

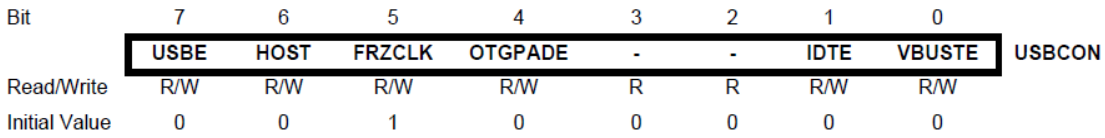


Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	SPEED		ID	VBUS	USBSTA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	1	0	1	0	

Obrázek 23 Ukázka registru USBSTA [9]

- SPEED – má funkci pouze v režimu USB Host. = 1 je-li připojeno zařízení s rychlostí Full speed, = 0 připojené zařízení je s rychlostí Low speed.
- ID – obsahuje výslednou hodnotu výběru typu zařízení.
- VBUS – má hodnotu 1, je-li připojeno napětí na USB sběrnici. Hodnotu 0 obsahuje, není-li napětí na USB sběrnici připojeno.

Typ zařízení lze také vybrat přímo v registru USBCON:



Bit	7	6	5	4	3	2	1	0	
	USBE	HOST	FRZCLK	OTGPADE	-	-	IDTE	VBUSTE	USBCON
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Obrázek 24 Ukázka registru USBCON [9]

- USBE – nastavením tohoto bitu se provede povolení celého USB řadiče.
- HOST – = 1 typ zařízení USB host, = 0 typ zařízení USB device.
- FRZCLK – nastavením se provede zastavení hodinové frekvence pro USB řadič. Vynulováním se naopak tato frekvence povolí.
- OTGPAGE – zapíná nebo vypíná hardwarovou kontrolu připojeného napětí na USB sběrnici.
- IDTE – povoluje nebo zakazuje přerušení při změně bitu ID.
- VBUSTE – povoluje nebo zakazuje přerušení při změna stavu (odpojení nebo připojení) napětí na USB sběrnici.

3.9 Kontrola připojení k USB sběrnici

Připojení k USB sběrnici lze detekovat pomocí přítomnosti napájecího napětí na USB sběrnici. Pro tento případ je v mikrokontroléru zaveden speciální obvod, který hlídá napěťové úrovně na vývodu VBUS. Porovnáváním napětí pomocí vnitřních komparátorů se nastavuje bit VBUS v registru USBSTA. Při detekci změny hodnoty bitu VBUS je nastaven bit VBUSTI a je-li také nastaven bit VBUSTE, je vyvoláno přerušení. Bit VBUSTI se nachází v registru USBINT:

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	IDTI	VBUSTI	USBINT
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Obrázek 25 Ukázka registru USBINT [9]

- IDTI – nastavením určuje změnu bitu ID.
- VBUSTI – nastavením signalizuje změnu bitu VBUS.

3.10 Koncové body

Mikrokontrolér je vybaven celkem sedmi koncovými body. Paměťový prostor koncových bodů může být rozdělen buď v jedné paměťové bance, nebo ve dvou paměťových bankách. Koncové body jsou uloženy v paměti DPRAM (Double Port Random Access Memory), která je zcela oddělena od paměti programu a paměti dat. Tato paměť má celkovou velikost 832 bajtů a má datovou strukturu paměti typu FIFO.

Všechny registry pro práci s koncovými body jsou v mikrokontroléru umístěny pouze jednou, tj. každý koncový bod nemá svou vlastní sadu registrů. Aby bylo možné určit, který koncový bod chceme ovládat nebo který chceme kontrolovat, je v mikrokontroléru přítomen registr UENUM:

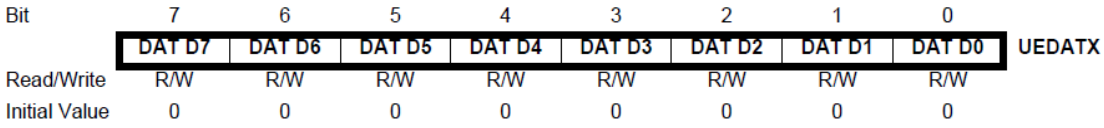
Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	EPNUM2:0			UENUM
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Obrázek 26 Ukázka registru UENUM [9]

- Zápisem čísla koncového bodu do tohoto registru dojde k nastavení všech registrů (UEDATX, UEINTX, UECFG0X, …).

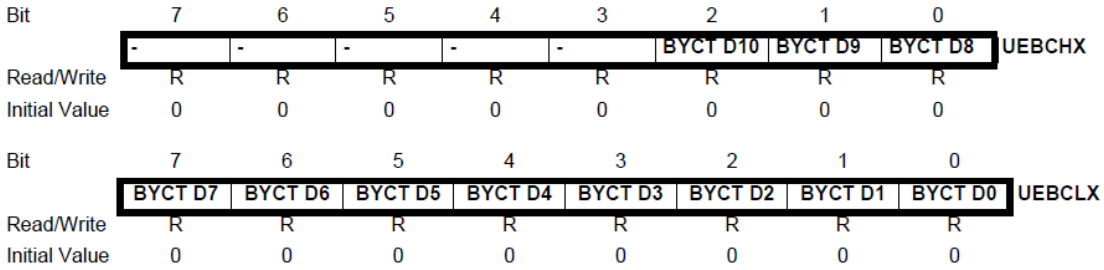
Pro výběr koncového bodu číslo 1 zapíšeme do tohoto registru číslo 1. Pro koncový bod 2 zapíšeme číslo 2 atd. Po zapsání čísla koncového bodu se všechny řídící a stavové registry (UEDATX, UEINTX, …) nastaví podle stavu vybraného koncového bodu.

Pro práci s pamětí FIFO je k dispozici několik registrů:



Obrázek 27 Ukázka registru UEDATX [9]

- Registr, který zprostředkovává přístup k datům. Při operaci čtení vrací data umístěná na začátku paměti. Operace zápisu ukládá data na aktuální pozici na konci paměti.

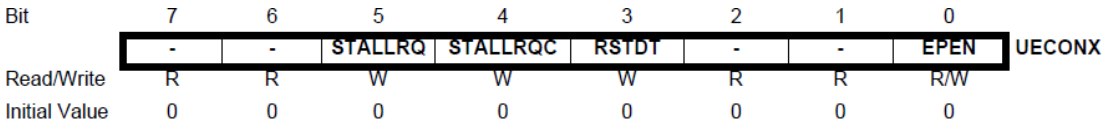


Obrázek 28 Struktura registrů UEBCHX a UEBCLX [9]

- Tyto dva registry obsahují počet bajtů uložených v paměti aktuálního koncového bodu.

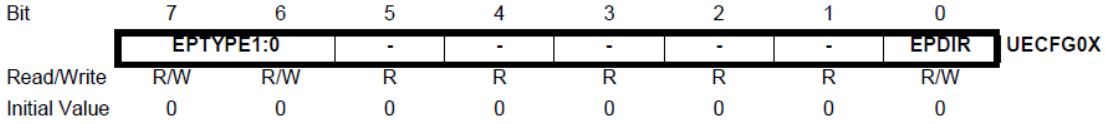
3.10.1 Alokace paměti koncového bodu

Aby koncový bod měl pro své operace vyhrazenou určitou velikost paměti, je nutné tuto paměť nejprve alokovat. K alokaci paměťového prostoru slouží tyto registry:



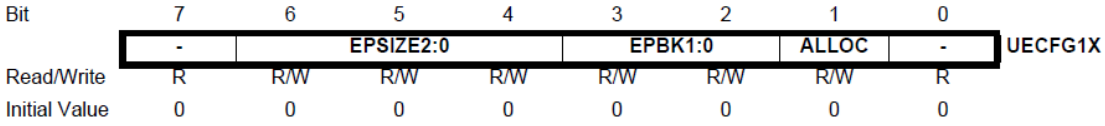
Obrázek 29 Struktura registru UECONX [9]

- STALLRQ – nastavením se vyšle STALL handshake, nuluje se hardwarově příjmem nového SETUP paketu.
- STALLRQC – nastavením se zastaví STALL handshake, vynuluje se okamžitě po nastavení.
- RSTDT – nastavením se provede reset typu dat na DATA0, vynulování se provede hardwarově okamžitě po nastavení.
- EPEN – nastavením se povolí koncový bod.



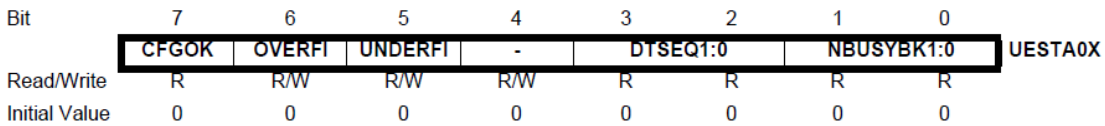
Obrázek 30 Struktura registru UECFG0X [9]

- EPTYPE1:0 – určuje typ koncového bodu – řídící, bulk, interrupt, izochronní.
- EPDIR – určuje směr koncového bodu. = 1 device → host; = 0 host → device.



Obrázek 31 Struktura registru UECFG1X [9]

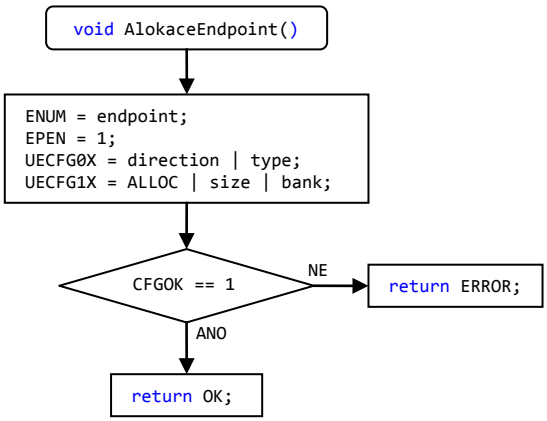
- EPSIZE2:0 – určují velikost koncového bodu.
- EPBK1:0 – v kolika bankách bude koncový bod rozložen.
- ALLOC – nastavením se provede alokace paměti koncového bodu. Vynulováním se dříve alokovaná paměť uvolní.



Obrázek 32 Struktura registru UESTA0X [9]

- CKGOK – signalizuje úspěšnost alokace koncového bodu. Hodnota 1 značí úspěšnou alokaci. Hodnota 0 značí, že alokace nebyla úspěšná.
- OVERFI – nastavení určuje, že došlo k přetečení koncového bodu při izochronním přenosu.
- UNDERFI – nastavení určuje chybu podtečení při izochronním přenosu.
- DTSEQ1:0 – určují PID paketu pro aktuální banku.
- NBUSYBK1:0 – obsahují číslo zaneprázdněné banky.

Celý proces alokace znázorňuje následující vývojový diagram:

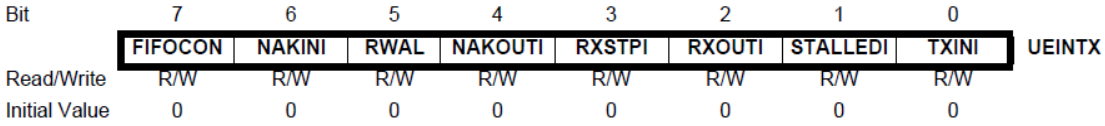


Obrázek 33 Vývojový diagram alokace paměti koncového bodu

Nejdříve je nutné vybrat koncový bod, který chceme alokovat. To provedeme zapsáním čísla koncového bodu do registru ENUM. Poté se nastaví bit EPEN. Zápisem do registrů UECFG0X a UECFG1X se nastaví parametry koncového bodu (směr komunikace, typ koncového bodu, velikost a počet paměťových bank, ve kterých má být uložen). Nastavením bitu ALLOC se provede alokace paměti. Testováním bitu CFGOK lze zkontrolovat úspěšnost alokace. Je-li tento bit nastaven, proběhla alokace v pořádku, je-li vynulován, alokace se nezdařila a je nutné změnit nastavení paměťové banky. Po správném nalezení nastavení banky lze toto nastavení pro danou velikost koncového bodu ponechat.

3.10.2 Operace čtení a zápisu koncového bodu

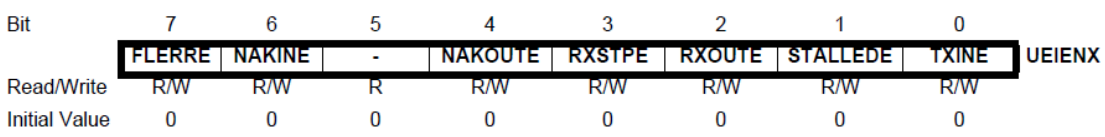
Pro práci s těmito operacemi je vyhrazen registr UEDATX a jeden stavový registr UEINTX:



Obrázek 34 Struktura registru UEINTX [9]

- FIFOCON – nastaven hardwarově při příjmu nebo odeslání dat. Vynulování způsobí vynulování paměti přijatých dat, nebo odeslání zapsaných dat. Nulováním zároveň dojde k přepnutí bank.
- NAKINI – nastaven hardwarově, když USB řadič odešle NAK handshake po USB sběrnici. Nulování je nutné provést softwarově.
- RWAL – kopíruje stav aktuální banky. Je-li tento bit nastaven, je možný zápis nebo čtení paměťové banky. Naopak je-li tento bit vynulován, není zápis nebo čtení umožněno.
- NAKOUTI – nastaven hardwarově, když je přijat NAK handshake z USB sběrnice. Nulování je nutné provést softwarově.
- RXSTPI – nataven hardwarově, když je přijat kompletní SETUP paket. Nulování je nutné provést softwarově, čímž dojde k potvrzení SETUP paketu.
- RXOUTI – nastaven hardwarově v případě, kdy byla obdržena nová data. Nulování je nutné provést softwarově.
- STALLEDI – nastaven hardwarově v případě, kdy USB řadič vyšle STALL handshake nebo došlo k chybě CRC kódu při izochronním přenosu.
- TXINI – nastaven hardwarově v okamžiku, kdy je aktuální banka prázdná a je možný zápis do banky. Nulování je nutné provést softwarově.

Tento registr obsahuje řadu bitů, které slouží jako příznaky přerušení, proto také existuje registr, který tato přerušení povoluje. Tímto registr je UEIENX:



Obrázek 35 Struktura registru UEIENX [9]

- FLERRE – povoluje přerušení při nastavení bitů OVERFI nebo UNDERFI pro izochronní přenos.
- NAKINE – týká se přerušení při nastavení bitu NAKINI.
- NAKOUTE – přerušení nastavením bitu NAKOUTE.
- RXSTE – povolí přerušení při příjmu celého SETUP paketu, čili při nastavení bit RXSTPI.
- RXOUTE – povolí vyvolání přerušení při nastavení bitu RXOUTI.
- STALLEDE – umožní přerušení nastavením bitu STALLEDI.
- TXINE – umožní přerušení nastavením bit TXINI.

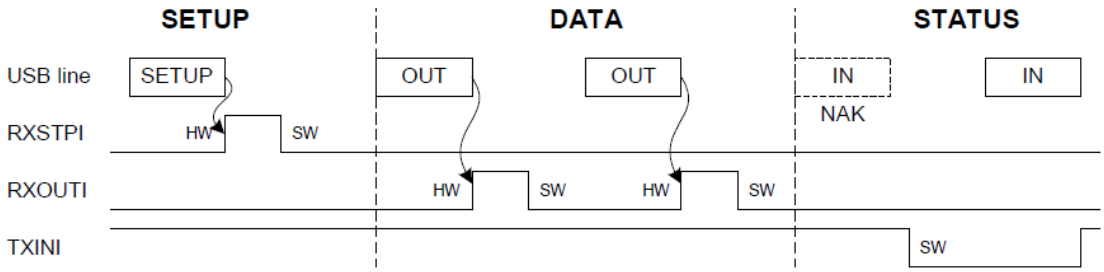
Operace čtení a zápisu se liší v závislosti na typu koncového bodu a na tom, v kolika paměťových bankách se koncový bod nachází.

Je-li koncový bod nastaven jako řídící, mají některé bity z registru UEINTX pro tento koncový bod jiný význam. Bit RXOUTI je nastaven v okamžiku přijetí paketu dat a jeho vynulování způsobí, že USB řadič vyšle potvrzení příjmu paketu a vyprázdní paměť koncového bodu. Bit TXINI je nastaven, když je možný zápis dat. Vynulování tohoto bitu zapříčiní odeslání dat a vyprázdnění paměti koncového bodu.

3.10.3 Čtení dat z řídícího kontrolního bodu

Operace čtení začíná vždy po přijetí SETUP paketu, kdy je třeba přečíst samotný SETUP paket. Toto čtení je velice jednoduché, protože v paměti je již uložen celý paket a tak stačí pouze osmkrát přečíst registr UEDATX. Těchto osm bajtů, což je velikost SETUP paketu, lze tedy číst ihned po nastavení bitu RXSTPI.

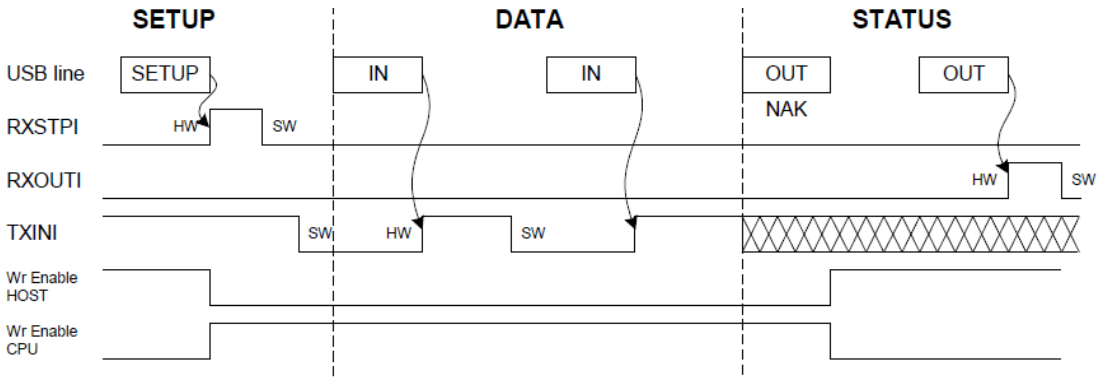
Ostatní data již nelze číst tak jednoduše, protože není jisté, kolik dat je nutno přečíst, a navíc data mohou mít větší délku, než je délka řídícího koncového bodu. Přijetí nového bajtu dat lze kontrolovat nastavením bitu RXOUTI a hodnotu těchto dat lze přečíst prostřednictvím registru UEDATX. Konec čtení dat je možné kontrolovat pomocí bitu TXINI. Když je tento bit nastaven, bylo USB řadičem automaticky odesláno potvrzení příjmu dat a byla tak přečtena všechna data. Podrobnější náhled na tuto problematiku nabízí následující obrázek.:



Obrázek 36 Čtení řídícího koncového bodu [9]

3.10.4 Zápis dat do řídícího koncového bodu

Stejně jako v případě čtení je vždy na začátku zápisu dat potřeba přijmout celý SETUP paket. Ten je opět signalizován nastavením bitu RXSTPI. Samotný zápis probíhá zapisováním hodnot do registru UEDATX. Zápis probíhá tak dlouho, dokud se nezapíší všechna data nebo pokud nebude velikost zapsaných dat rovna velikosti koncového bodu. V případě, kdy bude velikost zapisovaných dat rovna velikosti koncového bodu nebo vyšší, je potřeba vynulovat bit TXINI a čekat, dokud se opět nenastaví. To nastane v případě, kdy zapsaná data byla odeslána přes USB sběrnici. Kdyby došlo během procesu zápisu k nějaké chybě, zařízení USB host vyšle NAK handshake a tím se nastaví bit RXOUTI. Tento bit se nastaví i po zápise všech dat, kdy USB host vyšle ACK handshake. Přesnější přehled o zápisu dat ukazuje následující obrázek:

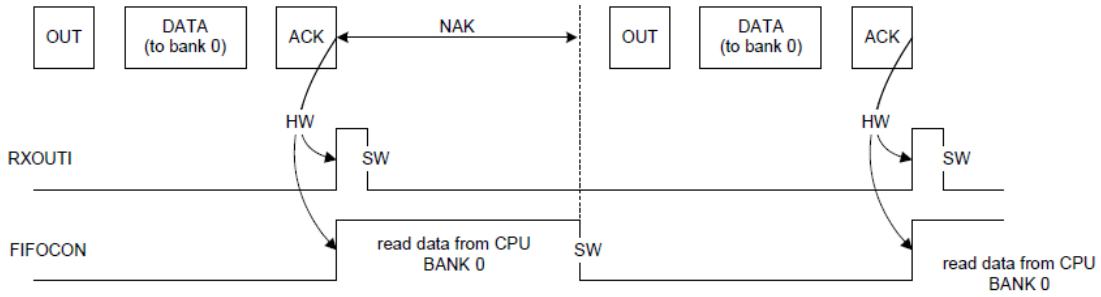


Obrázek 37 Zápis dat do řídícího koncového bodu [9]

3.10.5 Čtení dat z koncového bodu

Čtení dat z koncového bodu, který není nakonfigurován jako řídící koncový bod, se liší podle toho, v kolika bankách má uložena data.

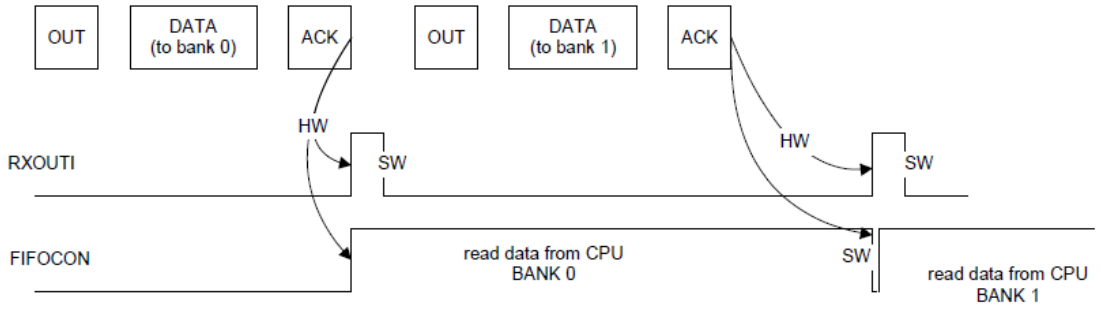
Následující obrázek ukazuje princip čtení dat z jedné paměťové banky:



Obrázek 38 Čtení dat z koncového bodu z jedné paměťové banky [9]

Přijetí dat signalizuje hardwarové nastavení bitu RXOUTI. Společně s tímto bitem je i nastaven bit FIFOCON. Bit RXOUTI lze ihned po zjištění jeho nastavení softwarově vynulovat. Pro čtení dat z paměťové banky koncového bodu lze využít i bit RWAL, který nelze využít při čtení a zápisu u řídícího koncového bodu. Nastavení tohoto bitu znamená, že lze stále z banky číst data. Naopak je-li tento bit vynulován, je již paměťová banka prázdná. Čtení dat se provádí pomocí registru UEDATX. Po přečtení všech dat uložených v paměťové bance lze vynulovat bit FIFOCON. Tento bit musí být vždy vynulován až po vynulování bitu RXOUTI.

Je-li paměťový prostor koncového bodu rozložen ve dvou paměťových bankách, je princip čtení složitější, jak naznačuje následující obrázek:



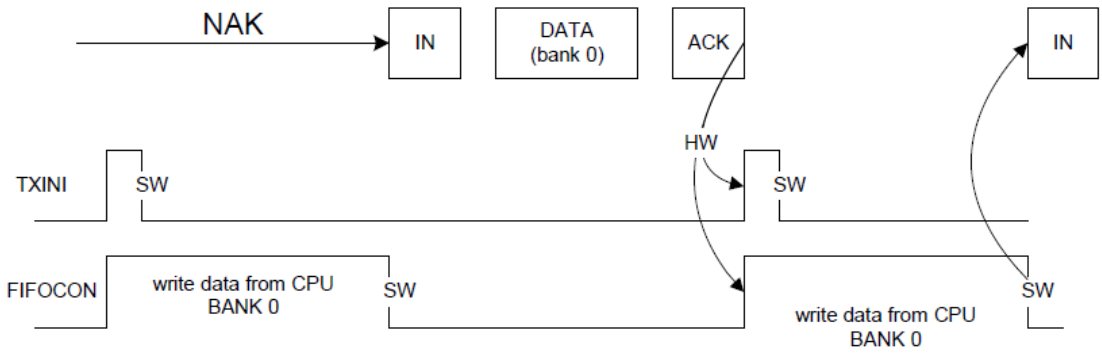
Obrázek 39 Čtení dat koncového bodu ze dvou paměťových bank [9]

Princip čtení ze dvou paměťových bank se od čtení dat z jedné paměťové banky liší pouze v mechanizmu přepínání mezi bankami. Pro účel přepínání bank slouží bit FIFOCON. Vynulováním tohoto bitu, když je aktuální banka již prázdná (RWAL = 0), dojde k přepnutí bank. Jsou-li v nově vybrané bance uložena nějaká data, dojde opět k nastavení bitu RXOUTI a bitu FIFOCON a je možné číst data z nově vybrané banky po dobu, kdy je nastaven bit RWAL.

3.10.6 Zápis dat do koncového bodu

Stejně jako v případě čtení dat z paměťového prostoru koncového bodu, který není nakonfigurován jako řídící koncový bod, je algoritmus zápisu závislý na tom, v kolika bankách je paměťový prostor rozložen.

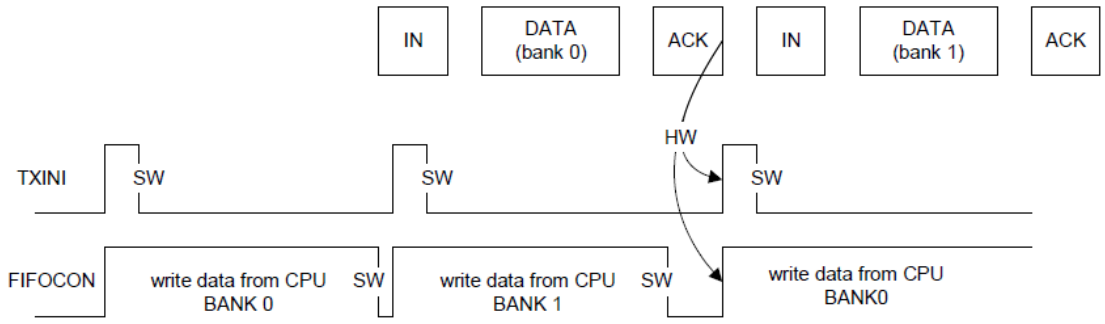
Princip zápisu do jedné paměťové banky je znázorněn na následujícím obrázku:



Obrázek 40 Zápis dat do koncového bodu v jedné paměťové bance [9]

Nastavení bitu TXINI a FIFOCON signalizuje možnost zápisu dat do paměťového prostoru koncového bodu. Bit TXINI lze ihned po detekci jeho nastavení vynulovat. Pro zápis lze opět využít bit RWAL, který je nastaven v okamžiku možnosti zápisu dat, a je vynulován, když je paměťová banka plná. Po zapsání všech dat se musí vynulovat bit FIFOCON, tím se začnou vysílat data na USB sběrnici. Bit FIFOCON musí být vynulován po vynulování bitu TXINI. Když se všechny data odešlou, dojde k nastavení bitu TXINI a FIFOCON.

V případě, že je paměťový prostor koncového bodu rozložen do dvou paměťových bank, se princip zápisu dat v mnohém neliší od předešlého případu. Rozdíl je v tom, že je opět nutné zavést mechanismus přepínání bank. Následující obrázek ukazuje proces zápisu:



Obrázek 41 Zápis dat do koncového bodu ve dvou paměťových bankách [9]

Nastavený bit TXINI a FIFOCON umožňuje zápis dat do aktuálně vybrané paměťové banky. Po dobu nastavení bitu RWAL lze zapisovat data do paměťové banky prostřednictvím registru UEDATX. V době, kdy dojde k vynulování bitu RWAL, musí dojít k softwarovému vynulování bitu FIFOCON. Tím dojde k přepnutí paměťových bank a také k začátku vysílání dat z předchozí paměťové banky po USB sběrnici. Přepnutím paměťových bank se zpřístupní zápis do nově vybrané banky, což signalizuje nastavení bitu TXINI a FIFOCON. Zápis se uskutečňuje stejným způsobem jako do předcházející banky.

3.11 Reset koncového bodu

Při obsluze požadavků USB hosta je v některých případech nutné provést reset daného koncového bodu. Reset všech koncových bodů se provádí pomocí jediného registru, kterým je UERST:

Bit	7	6	5	4	3	2	1	0	
	-	EPRST6	EPRST5	EPRST4	EPRST3	EPRST2	EPRST1	EPRST0	UERST
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

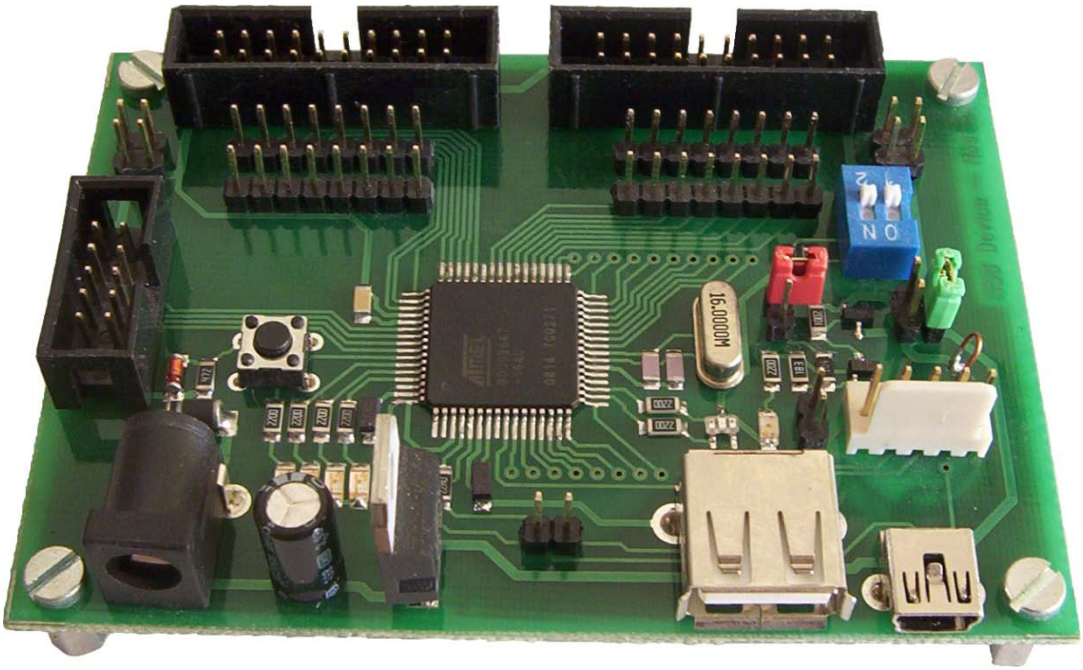
Obrázek 42 Ukázka registru UERST [9]

Jednotlivé bity tohoto registru představují reset daného koncového bodu. Tzn., je-li potřeba reset koncového bodu 0, nastaví se bit EPRST0 registru UERST. Pro koncový bod 1 se nastaví bit EPRST1 atd. Tento způsob je odlišný od způsobu výběru koncového bodu prostřednictvím registru UENUM.

Po výběru koncového bodu, resp. koncových bodů, je potřeba nastavené bity opět vynulovat. Tím se koncové body zpřístupní pro další operace.

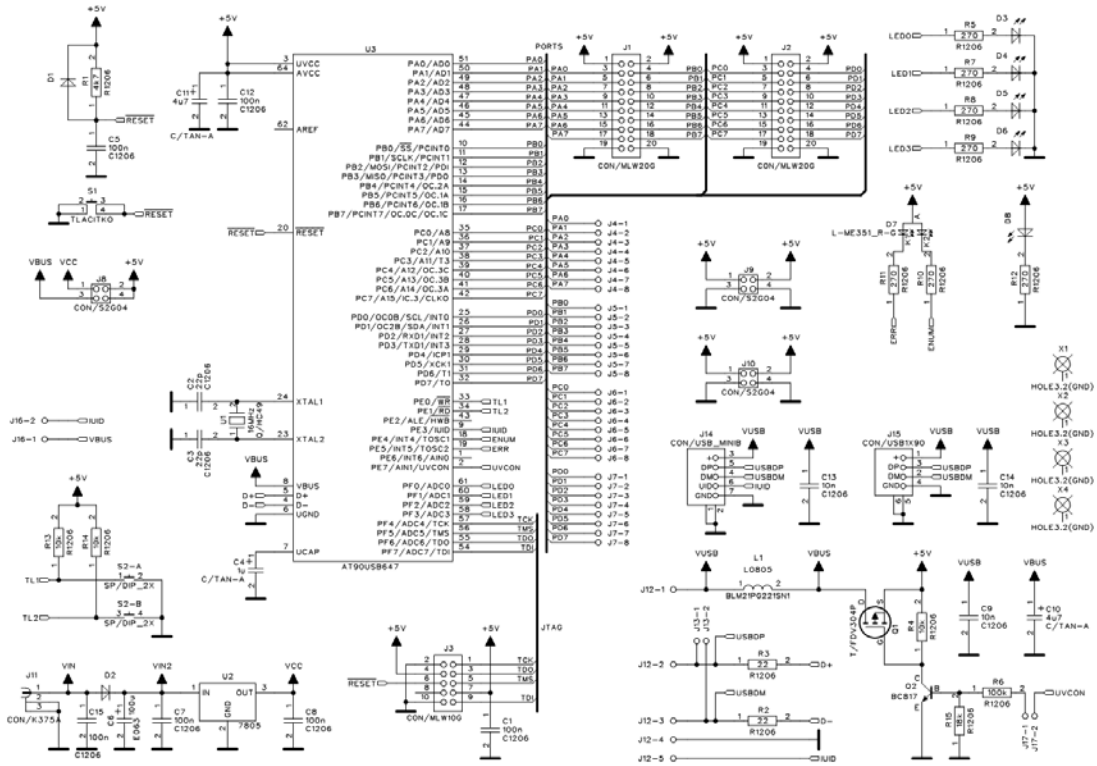
4 Vývojový kit

Za účelem vývoje zařízení komunikujícího po sběrnici USB a pro následné testování byl vyvinut jednoduchý vývojový kit, který obsahuje výše popsaný mikrokontrolér s několika okolními obvody. Schéma a stručný popis budou popsány v následující podkapitole.



Obrázek 43 Obrázek vývojového kitu

4.1 Schéma vývojového kitu



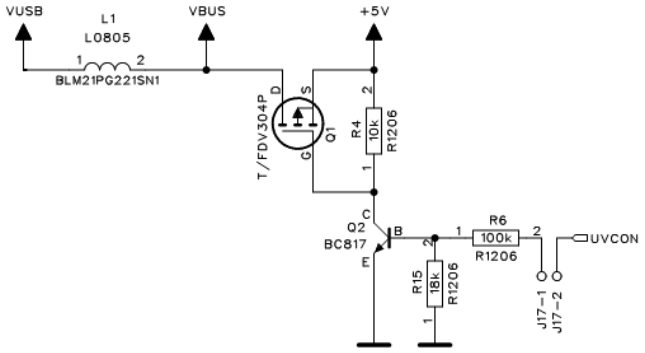
Obrázek 44 Schéma vývojového kitu

Zapojení mikrokontroléru U3 je optimalizované pro napájecí napětí +5V, proto je možné ho napájet přímo z USB sběrnice. Pro větší univerzálnost je možné pomocí konektoru J8 vybírat zdroj napájecího napětí. Jsou-li spojeny piny 1 a 2 konektoru J8, je vybráno napájení z externího zdroje napětí, spojením pinů 3 a 4 je vybráno napájení z USB sběrnice. Pro případ, že by bylo napětí z externího zdroje větší než +5V, je přidán stabilizátor napětí U2, který napětí zdroje stabilizuje na hodnotu +5V. Velikost napětí externího zdroje se tedy může pohybovat v rozmezí +8V až +30V. Obvod stabilizátoru je chráněn proti přepólování diodou D2, která je zapojená v sérii se stabilizátorem. Přítomnost napájecího napětí je signalizována LED diodou D8.

Vzhledem k faktu, že je mikrokontrolér napájen +5V, je možné k němu připojit zdroj hodinové frekvence s frekvencí až 16MHz. Tento zdroj je tvořen krystalem U1 s frekvencí 16MHz a dvěma keramickými kondenzátory C2 a C3. Obvod resetu pro mikrokontrolér je tvořen sériovým zapojením rezistoru R1 s kondenzátorem C5 a diodou D1. Reset je ještě možné provést ručně stiskem tlačítka S1. Pro účely ladění vytvářené aplikace byla k mikrokontroléru připojena řada signalizačních LED diod, těmito diodami jsou D3 až D6 a dvoubarevná dioda D7.

Pro potřeby připojení různých externích modulů je od mikrokontroléru vyvedena čtveřice vstupně/výstupních portů (PORTA, PORTB, PORTC, PORTD). Tyto porty jsou vyvedeny na dva konektory typu MLW20 a zároveň na čtyři lámací lišty. Vývojový kit lze připojit k USB sběrnici pomocí konektoru (J14) mini-B. Pro případné pozdější využití jako USB host lze zařízení připojit ke konektoru J15, který je typu A. Pro jiné využití lze použít konektor J12, kde jsou vyvedeny všechny vodiče potřebné k využití USB sběrnice, včetně vodiče IUID. Pro měření jsou vyvedeny datové vodiče D+ a D– USB sběrnice na konektor J13 a vodiče IUID a VBUS jsou vyvedeny na konektor J16.

Jelikož mikrokontrolér umožňuje pracovat i jako USB host a aby byl vývojový kit co možná nejuniverzálnější, obsahuje obvod pro řízení napájecího napětí na USB sběrnici. Tento obvod je znázorněn na následujícím obrázku:

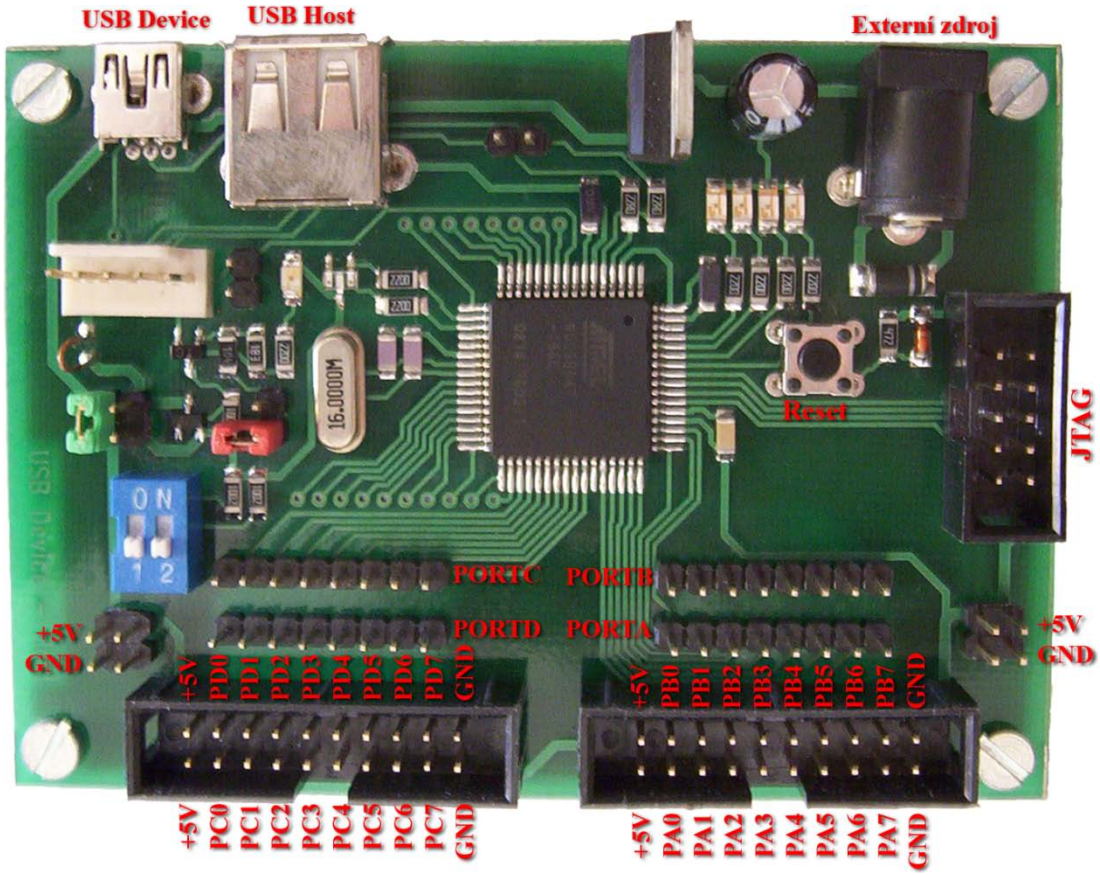


Obrázek 45 Napájení USB sběrnice

Obvod pracuje jako spínač řízený vývodem UVCON. Hodnota logické 1 na tomto vývodu způsobí otevření tranzistoru Q2. Jeho otevřením dojde k připojení vývodu G tranzistoru Q1 na zem, tím se mezi vývody G a S tranzistoru Q1 vytvoří dostatečně velké napětí k jeho otevření. Jelikož je tento tranzistor typu MOSFET, bude toto vytvořené napětí dostatečně veliké na to, aby tranzistorem mohl téct proud až 0,5A, aniž by došlo k velkému úbytku napětí na tranzistoru Q1. Aby se mohl tento obvod vyřadit z činnosti hardwarově, je přítomen konektor J17, jehož rozpojením dojde k odpojení obvodu od vývodu UVCON.

Mikrokontrolér je možné programovat prostřednictvím konektoru J3, kde je vyvedena sběrnice JTAG. Zapojení tohoto konektoru je totožné se zapojením na jiných vývojích kitech. Programování je možné také přes rozhraní SPI, které je vyvedeno na PORTB.

Následující obrázek ukazuje, kde jsou na vývojovém kitu vyvedeny všechny potřebné konektory včetně jejich popisu:



Obrázek 46 Zapojení konektorů na vývojovém kitu

Návrh plošného spoje vývojového kitu lze najít v příloze č. 1.

5 Firmware

Pro vyzkoušení práce s USB řadičem byly vytvořeny čtyři varianty obslužného softwaru. První variantou je USB klávesnice, neboli zařízení třídy USB HID. Dalším vyvinutým softwarem je virtuální sériový port, čili zařízení třídy CDC. Následovalo zařízení obsahující funkce obou předešlých zařízení. Umožňuje funkci jako USB klávesnice a virtuální sériový port v jednom zařízení. Poslední verzí firmwaru je nadstavba druhé varianty. Touto nadstavbou je druhý virtuální sériový port. Toto zařízení se chová jako dva sériové porty v jednou mikrokontroléru.

Všechny projekty byly napsány v programovacím jazyce C. Zdrojové texty k jednotlivým projektům lze najít na přiloženém CD, kde jsou uloženy jako soubory

projektů pro AVR studio v4.16. Každý projekt je složen z určitého počtu programátorských modulů. Mezi společné moduly patří soubory stddfn.f a Board.h.

Soubor Board.h je vždy spojen se zapojením vývodu mikrokontroléru na vývojovém kitu. Obsahuje např. makra pro nastavení směru vstupně/výstupních portů, počáteční hodnoty těchto portů, přiřazení názvu vývodů a makra pro manipulaci s jednotlivými vývody portů. Tato makra vypadají např. takto:

```
#define LED0 0 // Číslo vývodu, kde je připojena LED0
#define InitIoPortA() (DDRA = 0xFF) // Celý port je výstupní
#define InitPortA() (PORTA = 0) // Počáteční hodnota na portu je 0
#define SetLED0() (PORTF |= (1 << LED0)) // Nastav LED0
#define ClrLED0() (PORTF &= ~(1 << LED0)) // Vynuluj LED0
```

Soubor stddfn.h obsahuje názvy nových uživatelských datových typů a makra pro manipulaci s vícebitovými proměnnými, resp. proměnnými, které jsou větší než 8bitů. Příklad maker pro manipulaci s vícebitovými proměnnými:

```
#define LOW(x) (((uint8 *)&x)[0]) // Zpřístupní spodní byte 16b. proměnné
#define HIGH(x) (((uint8 *)&x)[1]) // Zpřístupní horní byte 16b. proměnné

#define LOW0(x) (((uint8 *)&x)[0]) // Zpřístupní nejnižší byte 32b. proměnné
#define LOW1(x) (((uint8 *)&x)[1]) // Zpřístupní druhý nejnižší byte 32b. proměnné
#define LOW2(x) (((uint8 *)&x)[2]) // Zpřístupní třetí nejnižší byte 32b. proměnné
#define LOW3(x) (((uint8 *)&x)[3]) // Zpřístupní nejvyšší byte 32b. proměnné
```

5.1 Zpracování základních činností USB sběrnice

Každé zařízení připojené ke sběrnici USB musí mít implementované základní funkce. Mezi tyto funkce patří např. nastavení adresy (SET_ADDRESS), zjišťování informací o daném zařízení pomocí deskriptorů atd. Pro tyto účely byly vytvořeny tři moduly:

- Modul Usb (soubory Usb.c a Usb.h) – obstarává obsluhu přerušení, inicializaci USB řadiče a zpracovává veškeré SETUP pakety.
- Modul UsbDescriptors (soubory UsbDescriptors.c a UsbDescriptors.h) – obsahuje veškeré deskriptory v podobě datových polí, které jsou uloženy v paměti programu. Zároveň obsahuje konstanty délek jednotlivých deskriptorů.
- Modul UsbEvents (soubory UsbEvents.c a UsbEvents.h) – obsahuje čtyři funkce, které jsou volány při zjištění určitých událostí spojených s USB sběrnicí. Tyto funkce jsou:

- UsbErrorSetup – volána při zjištění špatného SETUP paketu.
- UsbVbusConnect – došlo k připojení napětí na USB sběrnici.
- UsbVbusDisconnect – došlo k odpojení napětí na USB sběrnici.
- UsbEnumerationChange – volána při změně identifikace USB zařízení USB hostem.

Pro uložení aktuálního stavu USB sběrnice jsou použity dvě globální proměnné:

- usb_configuration – obsahuje 1 v případě, že bylo zařízení správně identifikováno USB Hostem. Hodnota 0 znamená, že zařízení není ještě zkonfigurováno. Konfiguraci zařízení je možné zkontrolovat ve funkci SET_CONFIGURATION, kterou vyšle USB host, v okamžiku, kdy byla dokončena identifikace USB zařízení. Tato proměnná je skrytá všem ostatním modulům, proto je použita funkce IsEnumeration, která vrací hodnotu této proměnné.
- usb_configuration_old – slouží jako záloha proměnné usb_configuration.

5.1.1 Inicializace USB řadiče

O inicializaci USB řadiče se stará funkce InitUSB(). Tato funkce je volána pouze jednou v hlavní funkci main.

```
// Inicializace USB řadiče
void InitUSB()
{
    // Inicializace PLL
    PLLCSR = (1 << PLLP2) | (1 << PLLP1) | (1 << PLLE); // Dělička 8x, povolení PLL
    while (!(PLLCSR & (1 << PLOCK))); // Čekání na ustálení frekvence PLL

    // Inicializace USB řadiče
    // Vnitřní ovládání ID, USB device, bez ovládání UVCON, zapnutí USB regurátoru napětí
    UHWCON = (1 << UIMOD) | (0 << UIDE) | (0 << UVCONE) | (1 << UVREGE);
    // USB device, "zmrazení" hodin pro USB, zapnout kontrolu napětí na VBUS
    USBCON = (0 << HOST) | (1 << FRZCLK) | (1 << OTGPADE) | (0 << IDTE) | (0 << VBUSTE);
    USBCON |= (1 << USBE); // Povolení USB řadiče

    UDCON = (0 << LSM) | (1 << DETACH); // Full Speed, odpojení interního pull-up rezistoru na D+
    UDIEN = (1 << SUSPE) | (1 << EORSME) | (1 << WAKEUPE) | (1 << EORSTE); // Povolení přerušení

    OTGCON = 0; // Konktrola OTG - žádná
    OTGTCON = 0; // Kontrola časů OTG - žádná
    OTGIEN = 0; // Zákaz přerušení OTG
    OTGINT = 0; // Vynulování příznaků přerušení OTG
}
```

V prvním kroku se nastaví jednotka PLL. Jelikož je na mikrokontrolér přivedena hodinová frekvence 16MHz a na vstup jednotky PLL je nutné přivést frekvenci 2MHz, musí se hodinová frekvence vydělit 8x. Po nastavení jednotky PLL je vhodné počkat na ustálení výstupní frekvence.

Dalšími nastaveními se zakáže hardwarový vývěr typu zařízení a nastaví se na pevno na USB device. Napájení mikrokontroléru je +5V, a proto je nutné zapnout vnitřní regulátor napětí. Dále je povolena hardwarová kontrola napětí na USB sběrnici. Rychlost zařízení je nastavena na Full speed a vnitřní rezistor na vodiči D+ je prozatím odpojen. Pro kontrolu stavu USB sběrnice je povolena řada přerušení.

5.1.2 Inicializace koncového bodu nula

Řídící koncový bod 0 je nutné inicializovat na několika místech v programu, proto byla inicializace umístěna do samostatné funkce InitEndpoint0.

```
// Inicializace USB, nastavení koncového bodu
void InitEndpoint0()
{
    USBCON &= ~(1 << FRZCLK); // Povolení hodin USB
    // Konfigurace koncového bodu
    UENUM = 0; // Endpoint 0
    UECONX = (1 << EPEN); // Povolení
    UECFG0X = 0; // Typ - Control Endpoint
    UECFG1X = (UECFG1X & (1 << ALLOC)) | 0x30; // Velikost 64B, One Bank
    UECFG1X |= (1 << ALLOC); // Alokace paměti
}
```

Alokace koncového bodu je možná pouze v případě, že je povolena hodinová frekvence USB řadiče. Pro konfiguraci koncového bodu 0 je nutné ho vybrat pomocí registru UENUM. Poté se koncový bod povolí, vybere se jeho typ, velikost a v kolika paměťových bankách bude uložen. V poslední řadě je možné spustit alokaci paměti koncového bodu. Kontrola správné alokace již není nutná, protože správné nastavení velikosti koncového bodu a počet bank byl nalezen již dříve.

5.1.3 Hlavní obsluha USB sběrnice

V hlavní funkci main je nutné neustále volat funkci USBTask, která se stará o kontrolu napájecího napětí na USB sběrnici a hlavně obstarává kontrolu SETUP paketů.

```
// Hlavní funkce pro obsluhu USB
void UsbTask()
{
    if (USBINT & (1 << VBUSTI)) // Došlo ke změně napájecího napětí na USB
    {
        USBINT &= ~(1 << VBUSTI); // Vynulování příznaku změny napětí na USB
        if (USBSTA & (1 << VBUS)) // Bylo napětí připojeno
        {
            InitEndpoint0(); // Inicializace koncového bodu
            UDCON &= ~(1 << DETACH); // Připojení interního pull-up na D+
            UsbVbusConnect(); // Zavolání funkce pro obsluhu připojení napájení
        }
        else // Napájení bylo odpojeno
        {
            usb_configuration = 0; // Zařízení již není nakonfigurováno
            // Zavolání funkce pro obsluhu změny konfigurace zařízení
            UsbEnumerationChange();
            UsbVbusDisonnect(); // Zavolání funkce pro obsluhu odpojení napájení
        }
    }

    UENUM = 0; // Výběr koncového bodu 0
    if (UEINTX & (1 << RXSTPI)) // Přišel SETUP paket
        UsbSetup(); // Zpracování SETUP paketu
}
```

Nejdříve se provede obsluha napájení USB sběrnice. Testem stavu bitu VBUSTI se zjistí, zda došlo ke změně stavu napětí na USB sběrnici. Je-li tento bit nastaven, došlo ke změně. Poté je možné tento bit vynulovat. Bit VBUS ukazuje aktuální stav napětí na USB. Nastavení tohoto bitu znamená, že došlo k připojení napětí, je-li vynulován, napětí bylo odpojeno. Došlo-li k připojení napětí, je možné inicializovat koncový bod 0, připojit interní pull-up rezistor pro výběr rychlosti zařízení a zavolat funkci UsbVbusConnect. V případě odpojení napětí se zruší konfigurace zařízení vynulováním proměnné usb_configuration a zavolají se dvě funkce UsbEnumerationChange a UsbVbusDisconnect.

Nakonec se v této funkci obslouží možný SETUP paket. Aby mohlo být zjištěno, zda je přijat SETUP paket, musí se nejprve vybrat koncový bod 0 prostřednictvím registru UENUM. Testem bitu RXSTPI se zjistí příchod SETUP paketu. Je-li tento bit nastaven, zavolá se funkce UsbSetup.

5.1.4 Obsluha SETUP paketů

O veškerou obsluhu SETUP paketů se stará výše zmíněná funkce UsbSetup. Tato funkce prochází všechny implementované možnosti USB funkcí, jako je např. nastavení adresy apod. Při nalezení shody se daná funkce zpracuje a funkce UsbSetup se předčasně ukončí pomocí příkazu return. Nebude-li nalezena žádná shoda, vyšle se STALL handshake a zavolá se funkce UsbSetupError. Protože SETUP pakety potřebuje zpracovat i modul, který bude implementovat vlastní

funkčnost zařízení, musí funkce UsbSetup jako jednu z možností funkce USB zavolat funkci SetupClassSpecific, které se pomocí parametrů předá celý SETUP paket. Pokud je SETUP paket korektní, tato funkce vrací hodnotu různou od nuly, hodnota 0 znamená chybu. Funkce SetupClassSpecific musí být umístěna v každé implementaci zařízení.

Další funkce, kterou musí mít implementována všechna zařízení, je funkce ConfigurateEndpointsClassSpecific, ve které se inicializují všechny koncové body daného zařízení. Tato funkce je volána při zjištění USB funkce SET_CONFIGURATION.

Neobsahuje-li daný SETUP paket žádná data, čili položka wLength je rovna nule, musí se pro potvrzení dat vyslat tzv. paket nulové délky. Ten se vyšle vynulováním bitu TXINI. Dokončení je signalizováno nastavením bitu TXINI.

Nese-li daný SETUP paket za sebou nějaká data určená pro USB zařízení, musí se načíst. Příjem dat lze zkontrolovat nastavením bitu RXOUTI. Jelikož je velikost koncového bodu dostatečně veliká a všechna přijímaná data se do této velikosti vejdou, stačí pouze daná data z koncového bodu přečíst prostřednictvím registru UEDATX. Po přečtení veškerých dat je nutné vynulovat bit RXOUTI a následně vynulovat bit TXINI pro vyslání potvrzení příjmu dat. Dokončení operace vyslání příjmu dat je možné kontrolovat prostřednictvím bitu TXINI, nastavení tohoto bitu určuje konec vyslání potvrzení.

Vyžaduje-li SETUP paket vyslání nějakých dat, např. vyslání určitého deskriptoru, je možné data do paměti koncového zapisovat, dokud se počet zapsaných bajtů nebude rovnat velikosti koncového bodu. Nedosahuje-li počet bajtů velikosti koncového bodu, stačí po zápisu všech dat vynulovat bit TXINI, čímž se data odešlou. Následně je potřeba počkat na nastavení bitu TXINI pro dokončení operace zápisu a na nastavení bitu RXOUTI, který signalizuje příjem potvrzení zápisu dat. Je-li velikost zapisovaných dat větší než je velikost koncového bodu, musí se do paměti koncového bodu zapsat jen tolik dat, která odpovídají velikosti koncového bodu. Poté se vynuluje bit TXINI a počká se na jeho nastavení, které nastane po odeslání zapsaných dat. Nyní se mohou zapisovat další data opět do velikosti koncového bodu a opět bude následovat vynulování bitu TXINI s čekáním na jeho nastavení. V případě, že

velikost koncového bodu je celým násobkem velikosti všech zapisovaných dat, je nutné na konci vyslat paket nulové délky, který se odešle vynulováním bitu TXINI, bez zápisu dat. Tento úkon je nutné provést z toho důvodu, že USB host neví, kolik dat se bude zapisovat, a odešlou-li se data s necelou délkou koncového bodu, je jasné, že je zápis u konce. V případě, že je délka dat stejná s délkou koncového bodu, musí se dát zařízení USB host najevo, že je již zápis u konce, a to se provede právě odesláním paketu nulové délky. Na konci operace zápisu je možné čekat na nastavení bitu RXOUTI, které ukazuje příjem potvrzení zápisu dat.

5.1.5 Obsluha přerušení

V rutině přerušení se obsluhují zejména události, kdy USB sběrnice přejde do stavu SUSPEND, WAKEUP a kdy se vyskytne reset na sběrnici.

Když USB sběrnice přejde do stavu SUSPEND, je vyvoláno přerušení nastavením bitu SUSPI. V tomto případě je nutné zastavit činnost jednotky PLL, zastavit hodinovou frekvenci USB řadiče nastavením bitu FRZCLK, uložit proměnou usb_configuration do proměnné usb_configuration_old, čímž se uloží stav konfigurace zařízení. Do proměnné usb_configuration se uloží číslo 0 a zavolá se funkce UsbEnumerationChange. Zakáže se přerušení při přechodu do stavu SUSPEND a povolí se přerušení při přechodu do stavu WAKEUP.

Přechod do stavu WAKEUP je signalizován nastavením bitu WAKEUPI. Zde je nutné provést opačné operace než při přechodu do stavu SUSPEND. Je nutné zapnout jednotku PLL, povolit hodinou frekvenci USB řadiče, obnovit hodnotu proměnné usb_configuration z proměnné usb_configuration_old a zavolat funkci UsbEnumerationChange. Zakázat přerušení při přechodu do stavu WAKEUP a povolit přerušení pro stav SUSPEND.

Poslední přerušení, které je nutné zpracovat, je přerušení při detekci resetu na USB sběrnici. Při zjištění tohoto stavu provede USB řadič reset koncového bodu 0, proto je nutné ho opět inicializovat funkcí InitEndpoint0. Při vyslání tohoto stavu nebude zařízení zkonfigurováno, proto se vynuluje proměnná usb_configuration a zavolá se funkce UsbEnumerationChange.

5.2 Projekt USB_CDC

Tento projekt vytváří USB zařízení, které se bude na operačním systému chovat jako virtuální sériový port. Tento projekt využívá pro svoji práci USB třídu CDC, jejíž celý popis lze najít v [10]. Třída CDC zahrnuje řadu podtříd. Konkrétně pro tento projekt byla vybrána podtřída ACM (Abstract Control Model), která je přímo určena pro sériové porty, modemy, apod. Model ACM umožňuje téměř veškerou funkčnost sériového portu, jako je např. nastavení přenosové rychlosti, nastavení typu parity, velikosti přenášených dat, ovládání výstupů a čtení stavu vstupů. ACM model umožňuje implementovat řadu protokolů. Ve většině případů jsou všechny protokoly spojené s různou definicí AT příkazů.

Tento model musí obsahovat celkem tři koncové body, které jsou rozmístěny do dvou rozhraní. Jedeno rozhraní obsahuje jeden koncový bod, který je typu interrupt a slouží jako informace o stavu vstupů. Toto rozhraní má nastavenou třídu na CDC, podtřídu nastavenou na ACM a protokol je nastaven na AT příkazy (V250). Druhé rozhraní obsahuje oba zbývající koncové body, které slouží pro přenos dat, proto má nastavenou třídu na CDC Data. Tyto koncové body jsou typu bulk. Jeden koncový bod slouží pro směr do USB zařízení a druhý pro směr z USB zařízení.

Pro ovládání třídy CDC byly do projektu přidány dva moduly k základním modulům. Tyto moduly jsou:

- Cdc (soubory Cdc.c a Cdc.h) – zpracovává SETUP pakety určené pro třídu CDC, obsahuje funkce pro čtení a zápis dat pro sériový port. V tomto modulu jsou potřebné funkce SetupClassSpecific a ConfigureEndpointsClassSpecific.
- CdcEvents (soubory CdcEvents.c a CdcEvents.h) – obsahuje sedm funkcí, které jsou volány v okamžicích, kdy nastanou určité stavy spojené se sériovým portem. Těla těchto funkcí jsou z počátku prázdná.
    - SetBreakEvent – volána v okamžiku příjmu funkce SetBreak.
    - ClrBreakEvent – volána v okamžiku příjmu funkce ClrBreak.
    - SetRts – volána v okamžiku nastavení výstupu RTS.
    - ClrRts – volána v okamžiku vynulování výstupu RTS.
    - SetDtr – volána v okamžiku nastavení výstupu DTR.
    - ClrDtr – volána v okamžiku vynulování výstupu DTR.
    - CdcNewData – volána v případě, že je zjištěn příjem nových dat.

Jelikož se zařízení chová jako sériový port, ve skutečnosti se nejedná o převodník z USB sběrnice na sériový port. Vytvořené zařízení se chová jako zařízení, které přímo využívá sériový port (poslaná data mikrokontrolér již nikam neposílá, ale přímo je zpracuje, jako odpověď na přijatá data může nějaká data odeslat). Toto řešení bylo použito z toho důvodu, že pro třídu CDC není potřeba psát žádný složitý ovladač, ale stačí použít ovladač nacházející se přímo v operačním systému.

Pro správnou funkci zařízení USB třídy CDC je v operačních systémech Windows potřeba instalace ovladače. Ve skutečnosti se jedná pouze o konfigurační soubor, který zařízení, jež má daný VID a PID, přiřadí ovladač nacházející se v operačním systému.

5.2.1 Funkce pro nastavování vlastností sériového portu

Jak bylo zmíněno, podtřída ACM zahrnuje řadu USB funkcí, které slouží k nastavení vlastností sériového portu. Jako všechny ostatní USB funkce i tyto jsou posílány prostřednictvím SETUP paketů, proto se s nimi zachází stejným způsobem jako v případě klasických USB funkcí, jejichž zpracování bylo popsáno v předchozí podkapitole.

Jednou z nejdůležitějších funkcí je funkce pro nastavení parametrů sériového portu. Tuto funkcí je SET_LINE_CODING. Pro tuto funkci vypadá SETUP paket takto:

Tabulka 9 SETUP paket funkce SET_LINE_CONDING

bmRequestType	bRequest	wValue	wIndex	wLength
00100001_B	20_H	0	Číslo rozhraní	7

Po tomto SETUP paketu následují data, která obsahují nové parametry sériového portu. Data mají tuto strukturu:

Tabulka 10 Struktura dat funkce SET_LINE_CODING

Název	Velikost	Popis
dwDTERate	32 bitů	Rychlost přenosu
bCharFormat	8 bitů	Počet stop bitů
bParityType	8 bitů	Typ parity
bDataBits	8 bitů	Počet přenášených bitů

Funkci SET_LINE_CODING vyvolá USB host v okamžiku, kdy se v programu, který ovládá daný sériový port, změní jeden z parametrů této funkce.

Opakem funkce SET_LINE_CONDING, která nastavovala parametry přenosu sériového portu, je funkce GET_LINE_CONDING, která vrací aktuálně nastavené parametry přenosu.

Tabulka 11 SETUP paket funkce GET_LINE_CODING

bmRequestType	bRequest	wValue	wIndex	wLength
10100001_B	21_H	0	Číslo rozhraní	7

Funkce GET_LINE_CODING vyžaduje zápis sedmi bajtů, které mají stejnou strukturu jako v případě funkce SET_LINE_CODING.

K ovládání výstupů na sériovém portu slouží funkce SET_CONTROL_LINE_STATE. Tu vyvolá USB host pokaždé, kdy je vynucena změna jednoho z výstupů. SETUP paket této funkce vypadá následovně:

Tabulka 12 SETUP paket funkce SET_CONTROL_LINE_STATE

bmRequestType	bRequest	wValue	wIndex	wLength
00100001_B	22_H	Stav výstupů	Číslo rozhraní	0

Tato funkce nevyžaduje žádná data, protože stav výstupů je nesen v proměnné wValue. První bit D0 proměnné wValue nese stav výstupu DTR a druhý bit D1 nese stav výstupu RTS. Ostatní bity jsou nastaveny do nuly.

Pro používání funkce SendBreak sériového portu je v modelu ACM zvláštní funkce, která se nazývá SEND_BREAK.

Tabulka 13 SETUP paket funkce SEND_BREAK

bmRequestType	bRequest	wValue	wIndex	wLength
00100001_B	23_H	Doba trvání stavu Break	Číslo rozhraní	0

I tato funkce nevyžaduje žádná data, protože potřebná informace je nesena v proměnné wValue. Je-li hodnota této proměnné $FFFF_H$, znamená to trvalé uvedení sériového portu do stavu Break. V tomto případě je volána funkce SetBreakEvent. Hodnota 0000_H znamená zrušení stavu Break. Nyní je volána funkce ClrBreakEvent.

Pro zaslání stavu vstupů sériového portu slouží koncový bod, který je umístěný v rozhraní s definicí třídy CDC. V projektu má tento koncový bod číslo 3. Zápisem dat do tohoto koncového se provede přesun stavu vstupů do cílové aplikace na počítači, ke kterému je USB zařízení připojeno. Zapisovaná data mají podobný formát paketu jako mají SETUP pakety, ale zapisovaná data se ukládají do zvláštního koncového bodu.

Tabulka 14 Formát rámce pro zaslání stavu zařízení

bmRequestType	bNotification	wValue	wIndex	wLength
10100001_B	20_H	0	Číslo rozhraní	2

Za těmito daty následují ještě dva bajty, ve kterých je uložen stav vstupů včetně stavu sériového portu. Formát těchto dat je následující:

Tabulka 15 Formát rámce stavu zařízení

Název bitu	Popis
D0 – bRxCarrier	Nahrazuje vstup DCD
D1 – bTxCarrier	Nahrazuje vstup DSR
D2 – bBreak	Určuje, je-li zařízení ve stavu Break
D3 – bRingSignal	Stav signálu Ring
D4 – bFraming	Chyba rámce
D5 – bParity	Chyba parity
D6 – bOverRun	Přijatá data mají být smazána
D7 – D15	Nastaveny na nulu

5.2.2 Operace čtení a zápisu

Operace čtení a zápisu z pohledu ovladače umístěného na počítači, ke kterému je připojeno USB zařízení, jsou realizovány pomocí dvojice koncových bodů, které jsou součástí rozhraní s třídou CDC Data. Tyto koncové body představují datové vodiče Rx a Tx sériového portu. Koncový bod pro směr do hostitelského počítače má číslo (adresu) 1, koncový bod pro směr z počítače má číslo 2. Čísla koncových bodů nejsou nijak závazná. Oba koncové body jsou umístěny ve dvou paměťových bankách.

Fakt, že přišla nějaká data, se kontroluje ve funkci CdcTask, kterou je nutné neustále periodicky volat v hlavní funkci main. Samotné zjištění příjmu nových dat se kontroluje pomocí bitu RXOUTI. Aby tento bit poskytoval skutečný stav koncového bodu, musí se daný koncový bod vybrat pomocí registru UENUM. Je-li tento bit nastaven, zavolá se funkce CdcNewData, do které si uživatel napíše vlastní obsluhu přijatých dat. Data je možné číst pomocí funkce CdcReadBuffer, která má dva parametry. Prvním parametrem je ukazatel, kam se mají data uložit, druhým parametrem je délka bajtů pro přečtení. Další funkcí pro čtení dat je funkce CdcReadByte, ve které se ale volá funkce CdcReadBuffer, která má nastaveny parametry tak, aby se načetl pouze jediný bajt. Obě funkce vrací hodnotu různou od nuly v tom případě, že se operace čtení zdařila, a vrací hodnotu nula v případě, že data nemohla být načtena,

např. zařízení není ještě správně identifikováno, proměnná usb_configuration je nastavena na hodnotu nula.

Samotný algoritmus čtení probíhá tak, že se čeká na nastavení bitu RXOUTI. Poté se provádí čtení dat z registru UEDATX do okamžiku, než se vynuluje bit RWAL. Po vynulování tohoto bitu je třeba vynulovat bit RXOUTI a bit FIFOCON. Vynulování bitu FIFOCON způsobí přepnutí bank a vyprázdnění paměti koncového bodu. Příjem dalších dat je možné kontrolovat bitem RXOUTI. Po nastavení tohoto bitu je možné opět číst data, dokud se nevynuluje bit RWAL. Tyto operace se provádí do té doby, než se načtou všechna potřebná data.

Pro odeslání dat lze použít funkci CdcWriteBuffer pro odeslání dat z paměti RAM a funkci CdcWriteBufferPgm, která odesílá data z paměti programu. Obě funkce mají dva parametry. Prvním je ukazatel na data, která mají být odeslána. Druhý parametr je délka dat. Funkce vrací hodnotu různou od nuly v případě, že data byla správně odeslána, hodnota nula znamená, že data nemohla být odeslána, protože zařízení není správně identifikováno USB hostem.

Algoritmus pro odeslání dat je trochu podobný s odesláním dat u řídícího koncového bodu. Pro zápis dat se musí vybrat správný koncový bod pomocí registru UENUM. Data se zapisují do registru UEDATX tak dlouho, dokud se nevynuluje bit RWAL. V okamžiku vynulování bitu RWAL je potřeba vynulovat bit TXINI a bit FIFOCON, čímž se provede vyslání dat po USB sběrnici a zároveň dojde k přepnutí paměťových bank. Pro další zápis dat je nutné počkat na nastavení bitu TXINI nebo RWAL, oba bity budou nastaveny ve stejný okamžik. Další data je opět možné zapisovat do okamžiku vynulování bitu RWAL. Tyto operace se provedou, dokud se nezapíší všechna potřebná data. Pokud byla délka zapisovaných dat stejná s délkou koncového bodu, je nutné na konci zápisu dat vyslat paket nulové délky, stejně jako u zápisu do řídícího koncového bodu. Ten se odešle tím způsobem, že počká na nastavení bitu RWAL a vynulují se bity TXINI a FIFOCON, čímž se odešle prázdný paket.

5.2.3 Testy a měření

Zařízení bylo testováno na operačních systémech Microsoft Windows XP SP2 a SP3, Microsoft Windows Vista, Linux (verze jádra 2.6.27.7-9). Měření probíhalo na operačním systému Microsoft Windows XP SP2 a bylo zaměřeno na zjištění přenosových rychlostí a na vytíženost mikrokontroléru.

Měření vytíženosti mikrokontroléru probíhalo tím způsobem, že se do programu přidaly instrukce pro nastavení a vynulování určitého výstupního portu, ke kterému byl připojen logický analyzátor. Nastavení portu se provedlo na začátku operace čtení nebo zápisu. K vynulování portu došlo v okamžiku, kdy mikrokontrolér čekal na odeslání nebo na příjem dat. Port byl tedy vynulován v okamžicích, kdy mikrokontrolér nevykonával žádné užitečné instrukce a mohl by vykonávat jiné operace.

Měření přenosových rychlostí, tedy rychlostí směrem z mikrokontroléru a do něj, probíhalo obdobně jako u měření vytíženosti. Pro měření byl také využit jeden výstupní port, ke kterému se připojil logický analyzátor. Port byl v programu nastaven v okamžiku začátku zápisu dat nebo v okamžiku zjištění příjmu nových dat a byl vynulován v okamžiku dokončení zápisu nebo čtení. Ze znalosti doby nastavení výstupního portu, tedy doby provádění operace čtení nebo zápisu a velikosti zapisovaných nebo čtených dat, lze snadno vypočítat přenosové rychlosti. Velikost zapisovaných a čtených dat byla nastavena na 4096 bajtů.

Naměřené hodnoty shrnuje následující tabulka:

Tabulka 16 Shrnutí naměřených hodnot

Přenosová rychlost při čtení (směr počítač → USB zařízení)	1,5Mb/s
Přenosová rychlost při zápisu (směr USB zařízení → počítač)	6,5Mb/s
Vytíženost mikrokontroléru při čtení	24%
Vytíženost mikrokontroléru při zápisu	98%

Provedená měření ukazují, že mikrokontrolér má nedostačující schopnosti pro maximální rychlost zařízení Full speed. Zajímavý je fakt, že rychlosti pro zápis a čtení jsou výrazně odlišné, ale z naměřené vytíženosti mikrokontroléru je patrné, že

mikrokontrolér dosahuje při zápisu své maximální rychlosti, ale při operaci čtení by mohla být rychlost přenášených dat mnohem větší. Tuto skutečnost lze přisuzovat např. ovladači sériového portu nebo faktu, že mikrokontrolér nahrazuje sériový port a rychlost 1,5Mb/s je na hranici maximální rychlosti sériového portu.

6 Závěr

Pro použití USB sběrnice ve vestavěných aplikacích je možné použít různé typy převodníků, které poskytují jednoduchý způsob implementace obvodů do zařízení. Mezi jejich nevýhody patří omezené možnosti použití. Druhým typem obvodů jsou mikrokontroléry s USB řadičem. Ty na rozdíl od převodníků umožňují široký rozsah možností typu zařízení, ale pro jejich implementaci je nutná hluboká znalost principů samotné USB sběrnice. Pomocí jednoho mikrokontroléru lze např. vytvořit několik typů zařízení najednou.

V průběhu práce jsem prostudoval velkou část principů funkce USB sběrnice, s jejíž pomocí se mi podařilo sestavit čtyři plně funkční zařízení. Tato zařízení byla důkladně testována a postupně byly odstraněny všechny chyby, které mi také do jisté míry napomohly k lepšímu porozumění funkce USB řadiče. Zároveň při vývoji aplikací s virtuálním sériovým portem jsem musel zvládnout techniku tvorby, resp. úpravy stávajících, konfiguračních souborů pro přiřazení souboru ovladače danému zařízení. Bez této úpravy by nebylo možné zařízení testovat na operačních systémech společnosti Microsoft.

Pro správnou funkčnost libovolného zařízení jsem vytvořil univerzální knihovnu pro základní zpracování požadavků USB sběrnice. Zařazením této knihovny, po úpravách deskriptorů, do projektu nového zařízení a rozšířením o funkčnost výsledného zařízení, lze dosáhnout rychlého vývoje zařízení.

Měření zařízení bohužel ukázalo, že přenosové rychlosti zdaleka nestačí pro maximální rychlost. Naměřené údaje dále naznačují, že za tyto výsledky nemusí nést vinu vytvořené zařízení. Příčina, jak bylo zmíněno, může být na straně ovladače použitého operačního systému.

Vyvinuté aplikace využívali jen část z celkových možností zvoleného mikrokontroléru. Vůbec nebyl použit typ zařízení USB host. Použití tohoto zařízení by již bylo nad rámec této práce. Z tohoto důvodu nejsou obvody na vývojovém kitu určené pro zařízení USB host využity. Jejich použití by bylo možné nalézt např. při realizaci zařízení umožňující připojení jednoduchých zařízení např. třídy HID.

Jedno z vyvinutých zařízení bylo použito při spolupráci se společností Steinel Technik při projektu videosenzoru. V tomto projektu rozšiřuje mikrokontrolér, naprogramovaný jako dva virtuální sériové porty, nedostatek vstupů a výstupu z procesoru Intel Atom, ke kterému bylo potřeba připojit např. relé nebo signalizační LED diody, ty by jiným způsobem nešli k procesoru připojit. Jako operační systém je zde použit operační systém Linux.

Seznam použité literatury

1. *Universal Serial Bus* [online]. [cit. 2009-08-24]. Dostupný z WWW: <http://www.usb.org>.

2. *Universal Serial Bus Revision 2.0 Specification* [online]. 2009 [cit. 2009-08-24]. Dostupný z WWW: <http://www.usb.org/developers/docs/usb_20_052709.zip>.

3. REDAKCE HW SERVERU. *USB 2.0 - díl 1* [online]. 2005 [cit. 2009-08-24]. Dostupný z WWW: <http://hw.cz/Rozhrani/ART1232-USB-2.0---dil-1.html>.

4. VOJÁČEK, Antonín. *Co se skrývá pod komunikací označenou jako USB OTG?* [online]. 2008 [cit. 2009-08-24]. Dostupný z WWW: <http://hw.cz/teorieapraxe/dokumentace/art2361-co-se-skryva-pod-komunikaci-oznacenou-jako-usb-otg.html>.

5. PŮHONÝ, Jan. *Vyšla specifikace USB 3.0* [online]. 2008 [cit. 2009-08-24]. Dostupný z WWW: <http://hw.cz/teorie-a-praxe/art2569-vysla-specifikace-usb-30.html>.

6. *Universal Serial Bus Revision 3.0 Specification* [online]. 2009 [cit. 2009-08-24]. Dostupný z WWW: <http://www.usb.org/developers/docs/usb_30_spec_052109.zip>.

7. *FTDI Chip* [online]. [cit. 2005-08-24]. Dostupný z WWW: <http://www.ftdichip.com>.

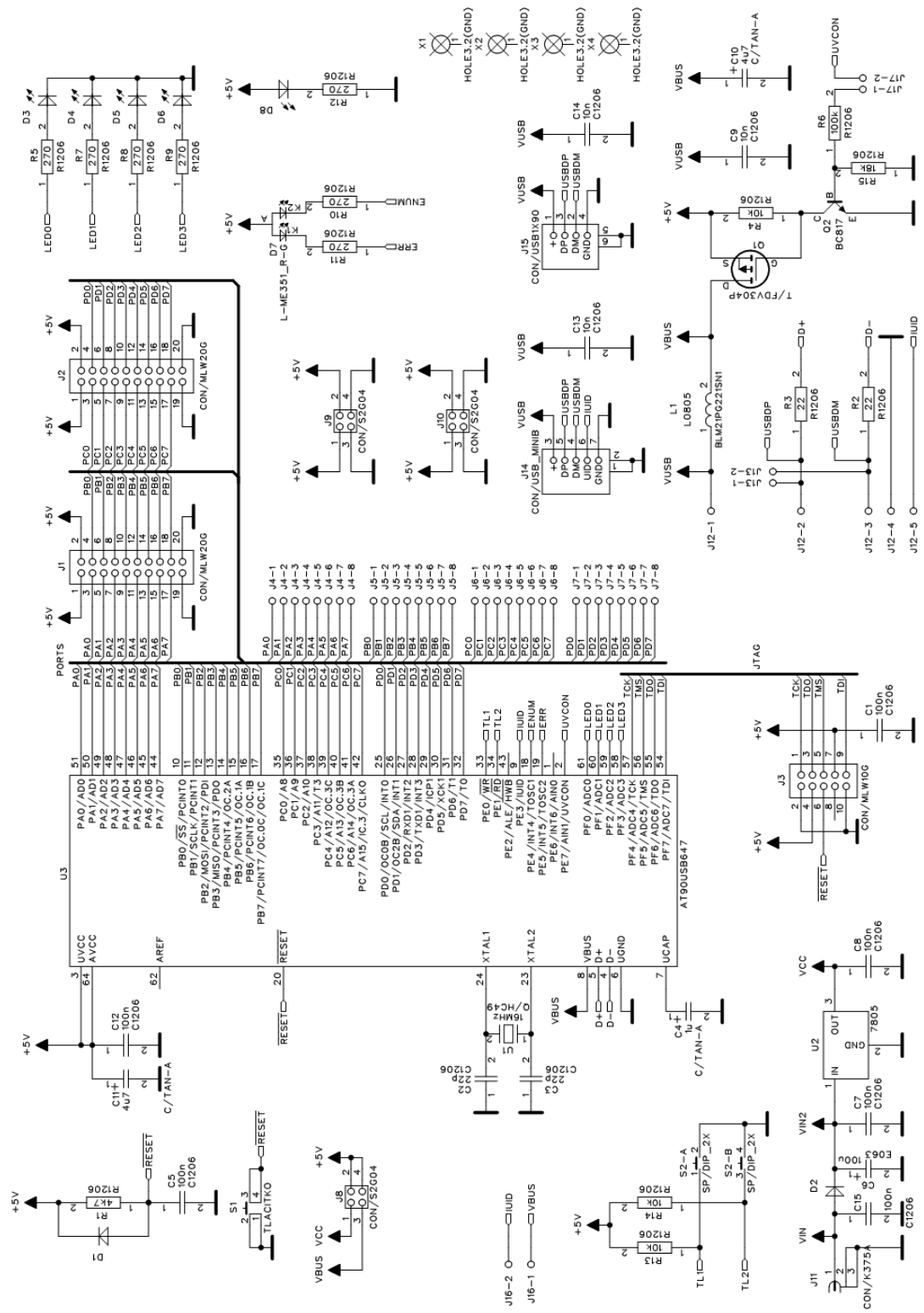
8. *Vinculum - Binding USB Technologies* [online]. 2008 [cit. 2009-08-24]. Dostupný z WWW: <http://www.vinculum.com>.

9. *Atmel Products - AT90USB647* [online]. 2009 [cit. 2009-08-24]. Dostupný z WWW: <http://www.atmel.com/dyn/resources/prod_documents/doc7593.pdf>.

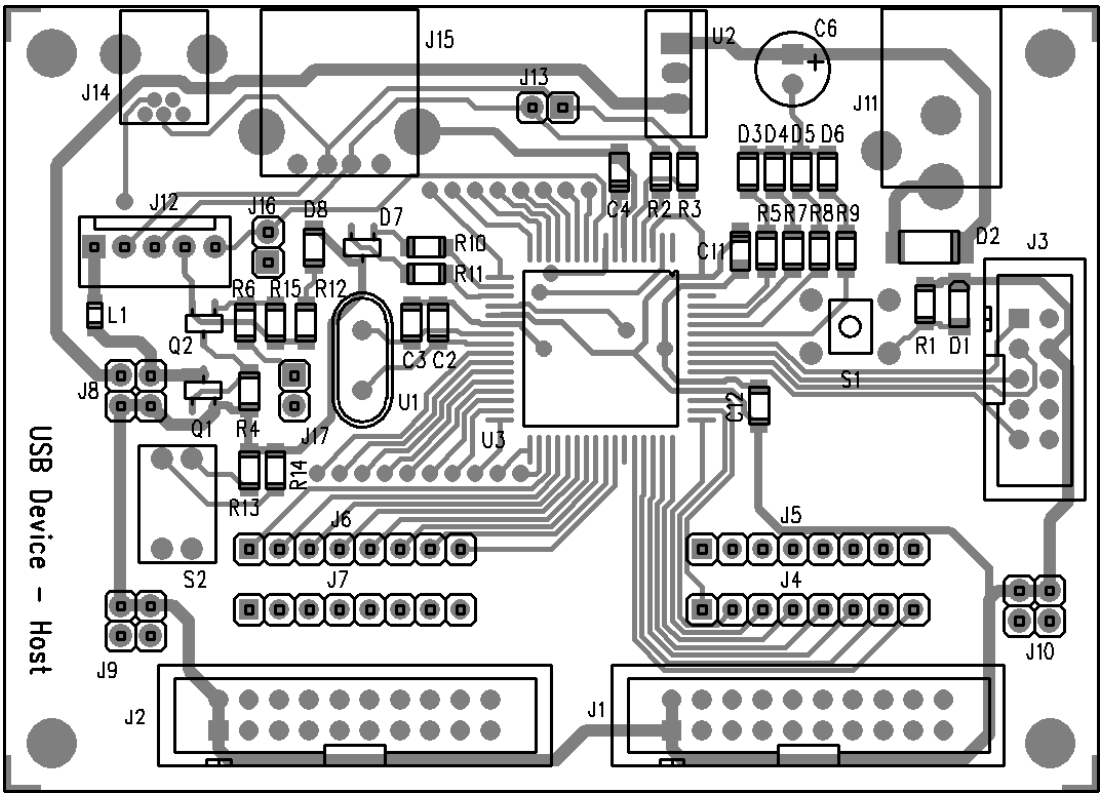
10. *Class definitions fo Communication Devices 1.2* [online]. 2007 [cit. 2009-08-24]. Dostupný z WWW: <http://www.usb.org/developers/devclass_docs/CDC1.2_WMC1.1.zip>.

Příloha

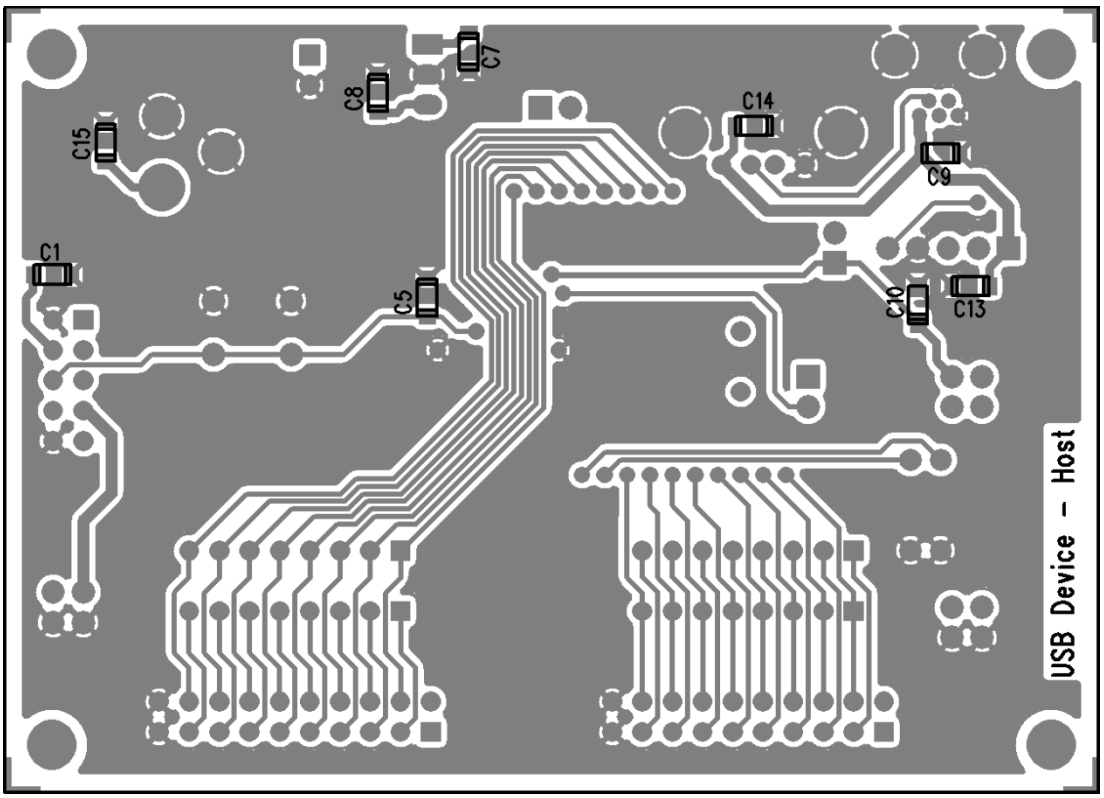
Příloha č. 1 – Schéma a návrh plošného spoje vývojového kitu



Obrázek 47 Schéma vývojového kitu



Obrázek 48 Návrh plošného spoje (horní pohled)



Obrázek 49 Návrh plošného spoje (spodní pohled)

Příloha č. 2 – CD:

Složka Texty – soubory PaarJ_USB Rozhrani_MH_2009.pdf a PaarJ_USB Rozhrani_MH_2009.docx obsahující vlastní text práce.

Složka Zdrojové kódy – obsahuje zdrojové kódy jednotlivých firmwarů

- HID_Keyboard – USB klávesnice.
- USB_CDC – virtuální sériový port.
- CDC_HID – kombinace USB klávesnice a virtuálního sériového portu.
- CDC_CDC – kombinace dvou virtuálních sériových portů.
- AVR Studio – obsahuje instalační soubory používaného vývojového softwaru AVR studio, včetně instalace WinAVR.

Složka Vývojový kit – obsahuje podklady pro tvorbu desky vývojového kitu, zejména soubory návrhového systému PADS 2005 SP2 a již vygenerovaná Gerber a Excelon data.