UNIVERZITA PARDUBICE
FAKULTA ELEKTROTECHNIKY
A INFORMATIKY

PROGRAMOVÁNÍ ŘÍDÍCÍCH APLIKACÍ POD OPERAČNÍM SYSTÉMEM LINUX

BAKALÁŘSKÁ PRÁCE

AUTOR PRÁCE: Michal Obešlo
VEDOUCÍ PRÁCE: Ing. Martin Hájek

2008

UNIVERSITY OF PARDUBICE
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATICS

PROGRAMING OF CONTROL APPLICATION UNDER LINUX OPERATING SYSTEM

BACHELOR WORK

AUTHOR: Michal Obešlo
SUPERVISOR: Ing. Martin Hájek

2008

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra informačních technologií
Akademický rok: **2007/2008**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Michal OBEŠLO**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Programování řídících aplikací pod OS Linux**

Zásady pro vypracování:

Navrhněte C++ třídu zapouzdřující komunikaci s obvody FT245 firmy Future Devices (FTDI), inc. Třída umožní událostmi řízenou komunikaci s obvodem prováděnou na pozadí aplikací. V práci proveďte shrnutí programového modelu operačního systému Linux a vlastností a programového modelu obvodů firmy FTDI.
Osnova práce:
* Architektura OS linux
* C/C++ Kompilátory
* Linux API
* Vícevláknové aplikace – vytváření vláken, synchronizace přístupu k datům, time management
* Obvody FTDI – technický popis, programový model
* Vytvoření C++ třídy zapouzdřující komunikaci s FTDI obvodem
* Vytvoření testovací aplikace

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

Matthew N., Stones R.: Linux Programujeme profesionálně, Computer Press, Praha 2001, ISBN: 8072265326

Matthew N., Stones R.: Computer Press, Computer Press, Praha 2000, ISBN: 8072263072

Introduction to GTK, dostupné online na http://www.gtk.org/

POSIX thread (pthread) libraries, dostupné online na http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html

Vedoucí bakalářské práce: **Ing. Martin Hájek**
Ústav elektrotechniky a informatiky

Datum zadání bakalářské práce: **30. listopadu 2007**

Termín odevzdání bakalářské práce: **16. května 2008**

Univerzita Pardubice Česká republika

L.S.

doc. Ing. Simeon Karamazov, Dr.
děkan

Poděkování

Rád bych poděkoval vedoucímu mé práce panu Ing, Martinu Hájkovi za jeho odborné rady a připomínky při tvorbě této bakalářské práce. Dále bych pak chtěl poděkovat panu Ing. Jiřímu Motlovi a panu Ing. Petrovi Šťastnému za cenné rady a podporu při vývoji.

Abstrakt

Bakalářská práce se zabývá návrhem třídy v jazyce C++, která umožní událostmi řízenou komunikaci s obvodem FTDI. Práce se dále zabývá návrhem testovací aplikace, která ověří funkčnost komunikace C++ třídy s obvodem FTDI pod operačním systémem Linux.

V programové části jsou popsány postupy vlastní komunikace, testování, připojení a manipulace s integrovaným obvodem FTDI pomocí spuštěného programu pod operačním systémem Linux.

Klíčová slova

Linux, Unix, FTDI, programování, C++, USB, vlákna, POSIX

Keywords

Linux, Unix, programming, C++, USB, thread, POSIX

Abstract

This bachelor thesis is about design and implementation of library for event-driven communication with the FTDI chip under Linux operating system. Implementation is in C++ language. There is a test application to demonstrate connection and communication with FTDI chip trough the library.

Obsah

Seznam obrázků

1 Úvod

V dnešní době se počítače třídy PC používají i pro řídící účely, kde ovládají externí zařízení, komunikují s integrovanými obvody nebo řídí výrobní procesy.

Pro externí komunikaci byl hodně rozšířený sériový port, ale nyní je v popředí hodně rozšířená sériová sběrnice USB (Universal Serial Buss). Díky unifikaci této sběrnice existuje mnoho variant přípojných zařízení.

Řízení a komunikace po této sběrnici je hodně rozšířená a podporovaná v operačním systému Windows a tudíž na něj existuje hodně ovladačů od různých výrobců a je na něj napsáno velké množství programu. To ale neznamená, že práce s USB je nepodporovaná v jiných operačních systémech např. Linux a že na jiné systémy neexistují ovladače. V dnešní době již výrobci nebo zdatní programátoři vytvářejí ovladače USB zařízení pro různé operační systémy. Tyto ovladače nabízejí ke komerčnímu i nekomerční použití.

Jednou z variant operačních systému je Linux, který se nyní dostává do podvědomí uživatelů a programátorů. Díky tomu, že je možné ho používat zdarma, tak se stává vhodnou volbou pro vývoj i programování řídících aplikací. Linux může být použit např. pro výuku ve školství, nebo jej využívají v malých firmách.

Náplní této bakalářské práce je vytvořit C++ třídu, která bude komunikovat s USB obvodem od společnosti Future Device (FTDI) a ověřit funkčnost a použitelnost USB řídících obvodů v operačním systému Linux.

V úvodní části se zaměřuji na popis architektury operačního systému Linux, popisu a metodice práce s vlákny standardu POSIX. Dále popisuji architekturu a vlastnosti sběrnice USB, integrovaný obvod a vlastnosti testovaného kitu. V hlavní části práce je popsán návrh obslužné třídy, programová realizace a výsledky testování.

2 Architektura operačního systému Linux

Dříve neměli uživatelé jinou možnost, než používat komerční systémy a komerční aplikace. Nemohli aplikace upravovat ani vylepšovat a museli striktně dodržovat licenční podmínky. Operační systém Linux tuto situaci změnil. Nyní si mohou uživatelé, správci systémů a programátoři zvolit volně dostupný operační systém s kompletní sadou nástrojů, aplikacemi a kompletním zdrojovým kódem.

Úspěch operačního systémů GNU/Linux potvrdila preciznost návrhu filozofie operačního systému Unix, z něhož Linux vychází. Spousta programů používaných v Unixu byla přenesena (portována) do Linuxu. Hlavními rysy filozofie operačního systému Linux je fakt, že se netváří jako celek (jádro + grafické uživatelské rozhraní jak je tomu u Windows), ale je tvořen z mnoha na sobě nezávislých částí, které mezi sebou komunikují (kernel, Windows manažer, X-Windows).

2.1 Jádro operačního systému Linux

Jádro systému (kernel) se zavádí při startu a zaručuje nejzákladnější operace (přístup k periferiím, řízení procesů, správu paměti atd.). Unix způsobil převrat tím, že oddělil jádro od zbytku systému a vypustil z něj zbytečné služby. I třicet let po svém vzniku je Unix a jeho modifikace na poli moderních operačních systému nezbytnou součástí. Za modifikace můžeme považovat systémy, jako jsou např. Linux, Solaris, FreeBSD, NetBSD, OpenBSD a další.

Hlavním úkolem jádra je ovládání prostředků počítače. Jádro umožňuje ostatním programům tyto prostředky používat. Jádro je z větší části napsáno v jazyce C a malou část kódu tvoří assembler.

V době kdy píši tuto práci je nejnovější verze stabilního jádra 2.6.25.9. Neexistuje pouze jedna větev vývoje monolitického jádra, ale současně probíhá vývoj i na mikrojádru, hybridním jádru a nano jádru.

Tyto verze se od sebe liší např.

- umístěným kódem běžícím v paměťovém prostoru (Unix, Linux, atd.)
- funkčnosti základních bloků jádra (např. mikrojádro - obsahuje jen správu paměti a správu řízení procesů)
- sdílená paměť jádrového paměťového prostoru

Příkladem monolitického jádra je např: Unix, Linux, MS-DOS, Windows 9x, ME, BSD, Solaris. Pro zkoumání a testy bylo použito standardní monolitické jádro z distribuce Linux - Ubuntu 6.06 LTS - Dapper Drake vydané v roce 2006.

V následujících několika odstavcích popisuji za jaké činnosti zodpovídá jádro Linuxu.

2.1.1 Zprostředkování přístupu k periferiím

Přístup k periferiím je velice choulostivou záležitostí. Díky filozofii přebrané z operačního systému Unix nemůže přistoupit více programů k jednomu zařízení současně. Pokud by byl takový přístup povolen, může dojít k nestabilitě, popřípadě pádu systému.

Proto byl vytvořen systém, kde může zároveň pracovat více uživatelů a běžet více procesů současně. Důležitou podmínkou je, aby jádro řídilo veškerý přístup k periferiím a procesy (spuštěné programy) nepustilo na hardware. Veškerý přístup na hardware obstarává jádro.

V operačním systému typu Unix je veškerý přístup k periferii sjednocen pod práci se soubory. Periferie např. typu Floppy, CD-ROM jsou pouze speciálním typem souboru.

Zápisem do takového souboru zapisujeme přímo na zařízení. Tato myšlenka dovoluje programátorům aplikací, které nejsou přímo určené pro práci se zařízeními to, že nemusí znát, nebo se učit API, ale stačí jim ovládat práci se soubory.

Veškerá zařízení mají svůj soubor s názvem umístěn v adresářové struktuře Linuxu v adresáři `/dev`.

Příkladem může být uživatel se superuživatelským oprávněním (root), který připojí zařízení CD-ROM jako soubor.

```
$ mount -t iso9660 /dev/scd0 /media/cdrom0
$ cd /media/cdrom
```

Připojili jsme jeho obsah jako souborovou strukturu pod adresář /media/cdrom. Teď se můžeme pohybovat po adresářích disku CD-ROM, ale pouze s omezeným přístupem čtení.

Tři důležité soubory zařízení jsou: `/dev/console`, `/dev/tty` a `/dev/nul`.

`/dev/console`

Zastupuje systémovou konzoli – terminál, na který se posílají důležité diagnostické zprávy a hlášení o chybách.

`/dev/tty`

Tento speciální soubor je logickým zařízením pro řídící terminál (klávesnici, obrazovku nebo okno) procesů, pokud ho má.

`/dev/nul`

Funguje podobně jako NUL v MS-DOSu. Vše co na něj pošleme, se ztratí. Při čtení z něj se vrací hned EOF.

O jednotlivá zařízení se starají ovladače, které byly v původním Unixu přímo zkompilovány do jádra, což znemožnilo např. vyměnit různý typ hardware, aniž by se provedla rekompilace s přidáním nebo odebráním ovladače pro daný typ zařízení. Na rozdíl od Unixu má Linux dynamické zavádění ovladačů zařízení, což značně usnadňuje jeho přenositelnost, např. při změně hardware není nutná reinstalace[1]. Většinou si Linux po spuštění systému detekuje hardware automaticky.

V dnešní době již vydává ovladače pod Linux spousta výrobců hardware. Díky tomu lze pracovat s hardwarem pomocí souborů a ovladačů, kterými disponuje jádro, v našem případě i s obvodem FTDI.

[1] reinstalace – přeinstalování operačního systému

2.1.2 Řízení procesů

Každý program v Linux se po spuštění stává procesem, který je plně v rukou jádra systému (kernelu). Kernel přidělí na základě výše jeho priority část paměťového prostoru a přidělí mu prostředky, je-li to vyžadováno.

Pro řízení procesů se uvnitř systémů využívá např. front, semaforů, mutexu.

Často se můžeme setkat s tím, že jeden program vytvoří v systému více procesů, což je obvyklé u démonů[2].

2.1.3 Plánování procesů

V Linuxu může běžet více programů zároveň, protože podporuje preemptivní multitasking. Tato vlastnost byla přidána za účelem lepšího řízení hardwarových přerušení a pro zlepšení podpory symetrického multiprocessingu[3].

2.1.4 Komunikace mezi procesy

Původní Unix podporoval komunikaci mezi procesy pouze pomocí signálů pipe (fronta, do níž jeden proces zapisuje a druhý proces zní čte).

Nástupci Unixu např. Linux, Solaris, FreBSD podporují mnoho dalších věcí pro komunikaci mezi procesy, jako je např. sdílená paměť, semafory atd.

2.1.5 Správa paměti

Správa paměti v operačním systému Linux je bez výhrady řízena jádrem. Jádro nastavuje adresní prostor pro aplikaci, nahrává soubory obsahující aplikační kód do paměti a poté předává řízení na pozici uvnitř programu, kde začíná jeho vykonávání. [3]

---

[2] démon – proces běžící na pozadí, např. Apache nebo tiskový server CUPS, někdy spuštěný přímo po startu systému

[3] symetrický multiprocesing – znamená spolupráci např. dvou naprosto shodných procesorů v rámci jednoho počítače [9]

Multitasking jádra umožňuje poskytovat uživateli iluzi současného běhu libovolného počtu procesů na počítači. Bez multitaskingu je počet procesů, které mohou na systému běžet zároveň, rovný počtu nainstalovaných CPU. To umožňuje každému programu se chovat jako by on byl jediný běžící a tím se chrání před kolizí s ostatními programy. Virtuální adresování umožňuje vytváření virtuálních částí paměti ve dvou rozdělených oblastech. Jedna bývá rezervována pro jádro (prostor jádra) a ostatní pro aplikace (uživatelský prostor). Procesor nepovoluje aplikacím, aby adresovaly paměť jádra, a tedy aby aplikace poškodila běžící jádro. [3]

Při nedostatku operační paměti se virtuální paměť ukládá na pevný disk, v Linuxu do swapovacího oddílu.

2.1.6 Bezpečnost

Jádro musí být navrženo tak, aby splňovalo víceuživatelský přístup, a zároveň musí být dbáno na bezpečnost uživatelských dat. Unix přišel jako první s pojmem uživatelé a skupiny. Hodně se v tomto systému dbá na přístupová práva např. k souborům, adresářům. Soubory mají tři základní parametry: čtení, zápis a spuštění.

Unix potažmo Linux má speciálního uživatele – správce (root). Tento uživatel může v systému provádět takřka cokoli, protože disponuje neomezenými právy. Díky právům si nemůže např. běžný uživatel přečíst soubory z jiného účtu, pokud mu to není povoleno. V některých případech vidí pouze adresáře.

2.1.7 Obsluha filesystému

Na rozdíl od Windows rozlišuje Linux malá a velká písmena, podporuje symbolické odkazy (jeden soubor může mít několik názvů). Filesystém podporuje vytváření speciálních souborů -“devices“ odkazujících na ovladače v jádře. Pomocí těchto odkazů přistupujeme na periferie.

Linux podporuje bezpočet různých filesystémů např. ext2, ext3, adfs, affs, coda, efs, msdos, vfat, hfs, hpfs, isofs minix, ncpfs, nfs, ntfs, qnx4, sysv, ufs, romfs, smbfs. Díky této podpoře mu nedělá problém spolupracovat s většinou ostatních operačních systémů.

3 Vlákna standardu POSIX

Filozofií vláken je, aby jeden program mohl dělat dvě věci zároveň. Případně aby jako současně zpracovávané vypadaly, nebo se v současnou dobu odehrály. Dříve měly Unixové systémy odlišné implementace vláken i jejich použití. Až s příchodem standardu, který vydal výbor IEEE POSIX, se vlákna v jednotlivých Unixových systémech sjednotila a zpřístupnila.

Vlákny nazýváme více větví prováděných z jednoho programu. Přesnější definice říká, že vlákno je „posloupnost řízení v procesu“. [3]

Pokud vytvoříme v procesu nové vlákno, bude mít toto vlákno svůj vlastní zásobník i lokální proměnné.

3.1 Výhody, nevýhody vláken

Vytváření nových vláken má oproti vytváření nových procesů některé výhody. Režie spojená s vytvořením nového vlákna je v podstatě menší než vytvoření nového procesu. (Ačkoli je systém Linux v porovnání s jinými operačními systémy při vytváření nového procesu hodně efektivní).

Někdy je nezbytné, aby program prováděl více operací najednou. Typickým příkladem kdy je takovýto přístup potřeba je počítání slov v dokumentu v reálném čase a současně umožnit uživateli dokument stále upravovat. V tomto případě může jedno vlákno pracovat s dokumentem a druhé vlákno může neustále aktualizovat proměnnou s počtem slov. Tohoto cíle lze daleko snáze dosáhnout vícevláknovou aplikací než pomocí více procesů.

Vícevláknové aplikace mají i své nevýhody. První velkou nevýhodou je větší složitost návrhu. V těchto aplikacích hrozí chyby při časování a neúmyslném sdílení proměnných. Druhou nevýhodou je ladění vícevláknových aplikací oproti jedno vláknovým aplikacím.

Linux implementuje vlákna pomocí systémového volání „clone“. Toto volání používají pouze knihovny pracující s vlákny.

Následující popis je proveden dle. [3] [12]

3.1.1 Ověření podpory vláken

Pokud chceme používat vlákna, je rozumné si ověřit, zda náš systém opravdu podporuje vícevláknové aplikace standardu POSIX.

Tuto kontrolu můžeme provést dvěma způsoby. Jednodušším a složitějším. Obtížnější způsob spočívá v prohledávání hlavičkových souborů, jako je limits.h, unistd.h a (pro Linuxuvé systémy charakterističtější) feature.h.

Jednodušší způsob spočívá v napsání krátkého programu, jeho přeložení a kontrole výstupu. Ověřovací program nalezneme v knize Linux začínáme programovat na straně 373. [3]

3.2 Práce s vláknem

Pro práci s vlákny můžeme použít následující funkce pro tvorbu vlákna a jeho zrušení:

```
int  pthread_create(pthread_t * thread, pthread_attr_t * attr,
 void * (*start_read)(void *), void * arg);
```

- funkce vytvoří nové vlákno, které bude vykonávat funkci `start_read`, což je funkce akceptující jeden parametr typu `void *`. Na adresu `thread` je uložen identifikátor vlákna a jako atributy vlákna můžeme uvést `NULL` pro implicitní hodnoty.

```
void pthread_exit(void *retval);
```

- tato funkce předčasně ukončí vlákno, ze kterého byla funkce zavolána.

Vlákno se také ukončí, skončí-li funkce `start_read`. V obou případech se předává návratový kód.

```
int pthread_join(pthread_t th, void **thread_return);
```

- funkce čeká na ukončení vlákna `th`. Na adresu `thread_return` je uložen návratový kód vlákna.

Při překladu programu používající vlákna je třeba program slinkovat s knihovnou lpthread. Další příklady práce s vlákny jsou uvedeny v knize [3] nebo na internetových stránkách. [5]

Následující popis je proveden dle. [3] [4] [5]

3.3 Mutex a semafory

Při práci s vlákny nastanou případy, kdy jedno vlákno nesmí použít stejná sdílená data ve stejném okamžiku jako druhé vlákno. Může nastat nekonzistence dat. Tomu se zabraňuje tím, že práci s danými daty může vykonávat jen jedno vlákno a druhé čeká, až první práci ukončí. Takováto sekce se nazývá MUTEX. Ta má dva stavy uzamčený a odemčený. Semafory jsou obdobné jako MUTEX, ale je možné jimi docílit více vláken v sekci.

Podrobnější informace o tom jak pracovat s mutexem a semafory je uvedeno v knize [3].

4 Sběrnice USB

Sériová sběrnice USB (Universal Serial Buss) se v posledních letech stala samozřejmou součástí osobních počítačů. Toto rozhraní nahrazuje dříve používané způsoby připojení (sériový, paralelní port, PS/2 apod.) pro běžné druhy periferií i tiskárny, faxy, zvukové karty atd., ale i pro přenos z čteček paměťových karet, externích disků, externích vypalovaček.

Data se tedy po této sběrnici přenášejí bit po bitu podobně jako u RS232. Podstatná změna nastává v tom, že sériový port mohl využívat vždy jen jedno zařízení, naproti tomu USB až 127 zařízení.

Základní parametry sběrnice USB:

- sériové rozhranní
- rychlost 1.5, 12, 480 Mb/s
- připojení zařízení až na vzdálenost 5 m
- možnost napájení z konektoru
- až 127 připojených zařízení
- podpora plug&play

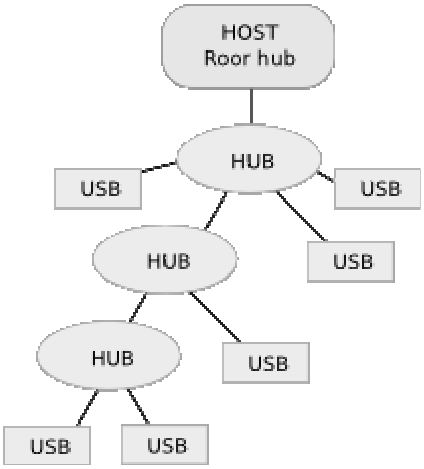
4.1 Architektura USB

Architekturu USB můžeme rozdělit do třech částí:

1. Hostitelský hardware a software je část uložená v počítači. Má za úlohu zprostředkovat kontakt mezi softwarem a USB periferiemi tak, aby se výstupně - vstupní systém jevil jako jeden celek a aby se software nemusel zabývat konkrétními záležitostmi týkající se připojení jednotlivých periférii.
2. Druhou úrovní jsou takzvané huby nebo rozbočovače. Počítače, které disponují rozhranním USB, mají obyčejně dva, případně více USB konektorů. K počítači můžeme připojit až 127 zařízení USB. Abychom takové připojení mohli realizovat, potřebujeme zvětšit počet přípojných míst pomocí hubu.

Hub je samostatné zařízení, nebo je součástí jiného zařízení. Připojuje se zpravidla na port USB a má zpravidla 4 až 7 USB konektorů, do kterých je možno připojit koncové zařízení nebo další USB huby. Mimo toho, že huby zvyšují počet přípojných míst, mají za úkol také řídit komunikaci s připojenými, odpojenými zařízeními. Což znamená identifikovat zařízení, přidělit mu identifikátor, pomocí kterého ho oslovuje řadič USB.

3. Na huby jsou připojená jednotlivá zařízení USB. Ty musí vykazovat určitou inteligenci v tom, že umí identifikovat, popsat svoji konfiguraci, tak aby bylo možné pro zařízení přiřadit správné ovladače a použít je. To umožňuje připojovat a odpojovat zařízení přímo za běhu operačního systému (hot plug a hot unplug).



Obrázek 1 - Topologie sběrnice USB

4.2 Softwarové a systémové vlastnosti USB

- Automatická identifikace periferií, automatické mapování funkce a konfigurace driveru
- Dynamicky připojitelné a konfigurovatelné periférie
- Automatická indikace připojení a odpojení periferie
- Současná funkce více zařízení připojených na společnou USB sběrnici
- Podpora zařízení s mnoha sdruženými funkcemi
- Jednoduchý protokol s vysokou efektivitou

- Robustnost - zpracování chyb je součástí protokolu
- Široký rozsah délky paketů, umožňující optimalizaci využití protokolu
- Garantované šířky pásma pro zařízení, které jej vyžadují (telefon, audio atd.)
- Možnost využití celé šířky pásma jedním zařízením
- Dynamické přidávání a ubírání zařízení při jejich připojení/odpojení
- Podpora pro identifikaci vadných zařízení, indikace chyb přenosu a možnost jejich korekce.
- Podpora synchronních i asynchronních přenosů
- přenosové rychlosti od 1,5 Mbit/s do 480 Mbit/s
- možnost připojení zařízení na relativně velkou vzdálenost (až 5 m)
- možnost napájet zařízení přímo z konektoru (odebíraný proud je běžně 100 mA, lze však odebírat proud až 500 mA)
- Možnost připojení velkého počtu zařízení (při použití HUBů až 127)
- Podpora Plug&Play (připojování a odpojování zařízení za provozu)
- Podpora operačními systémy Windows 98/2000/Me/XP (také Linux, MAC, OS-8, OS-9, OS-X). [4]

5 Integrované obvody

Kromě velkého množství kompletních, funkčních periférií k PC je na trhu i široká nabídka integrovaných obvodů pro použití s USB sběrnicí, od jednoúčelových převodníků (např. USB na RS-232, FT232, FT245BM) až po jednočipové mikrokontroléry se zabudovaným USB rozhraním.

Tyto obvody nabízejí např. společnosti Future Technology Devices (FTDI), Prolific Technology, SigmaTel, Texas Instruments, Elan, Cygnal, Silicon Laboratories a Semtech. Obvody FTDI jsou v současnosti pravděpodobně nejznámější dostupná USB rozhraní na českém trhu, kde je distribuuje společnost ASIX s.r.o. [4]

Pro ověření funkčnosti komunikace integrovaného obvodu pod operačním systémem Linux byl vybrán obvod FTDI 245BM od společnosti FTDI. Hlavní důvod je ten, že již dříve byl tento obvod testován pod operačním systémem Windows a pracoval bez sebemenších komplikací.

5.1 Integrovaný obvod FT245BM

Převodník FT245BM umožňuje konverzi mezi USB a paralelní 8 bitovou vstupně-výstupní branou na bázi FIFO.

Obrázek 2 - Pouzdro obvodu FT245BM

Význam koncovek v typovém značení je stejný jako v případě FT232. To znamená, že obvody jsou identické a liší se pouze typem pouzdra.

5.1.1 Parametry

- Přenosová rychlost až 1MB/s (ovladače FTDI D2XX)
- Hardwarové řízení toku dat
- Vstupní buffer 128 Bytů, výstupní buffer 384 Bytů
- Pouzdro LQFP-32 nebo QFN-32
- Kompatibilita s USB 1.1 a USB 2.0
- Napětí napájecí (4,35V - 5,25V)
- Možnost připojení k 5V i 3,3V logice
- Integrovaný obvod Power-On-Reset

5.1.2 Uplatnění

- Řízení pomocí USB
- MP3 přehrávače
- USB rozhranní digitálních kamer
- USB rozhraní čtečky paměťových karet
- atd.

6 Vývojový kyt

Pro účely testování mi byl zapůjčen školní vývojový kyt s integrovaným obvodem FTDI 245MB od společnosti PK-Design. Tato společnost nabízí vývojové kity od různých výrobců např. Atmel, Xiling a FTDI.

Testovací balení obsahovalo:

1x základnovou desku

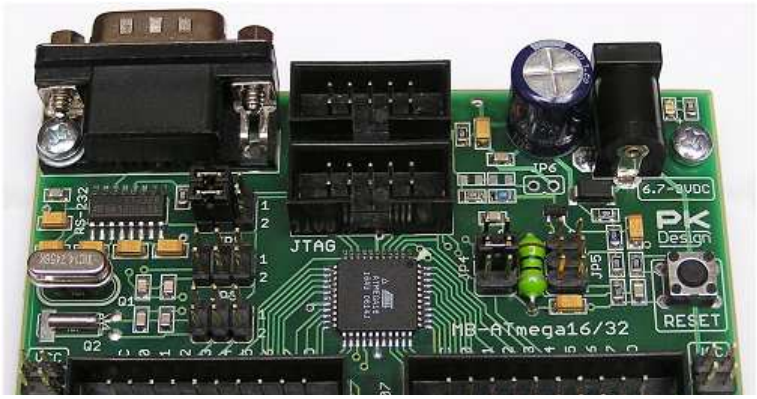
1x modul FTDI

1x mini modul spínačů

1x mini modul led diod

6.1 Základnová deska

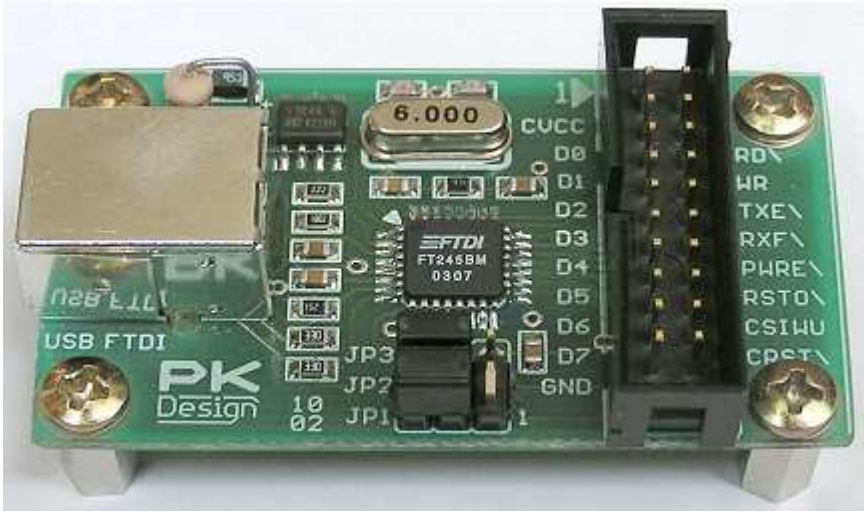
Základová deska MB-ATmega16/32 je jednou z hlavních částí vývojového a výukového modulárního systému MVS. Obsahuje RISC-ový mikrokontroler vývojové řady AVR firmy Atmel s označením ATmega16(L)/32(L)-16AU(8AU). Deska je tedy určena k vývojovým či výukovým účelům v oblasti RISC-ových mikrokontrolerů. Výhodou těchto výrobků je nízká cenové kategorie s maximálním výkonem 16 MIPS. [6]



Obrázek 3 - Základnová deska MB-ATmega 16/31

6.2 Modul FTDI s obvodem FTDI 245MB

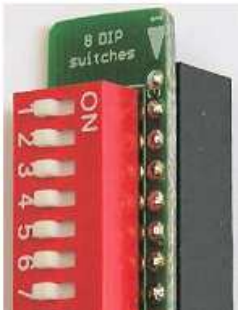
Modul je založen na obvodu FT245BM firmy FTDI (obousměrný konvertor USB-FIFO) a obsahuje vše potřebné pro komunikaci základové desky vývojového systému MVS s osobním počítačem přes standardizované rozhraní USB. Obvod FT245BM je založen na principu FIFO. [6]



Obrázek 4 - Modul je založený na obvodu FT245BM

6.3 Mini modul přepínačů

Mini modul přepínačů se zasouvá do základové desky vývojového kitu přímo, obsahuje 8 DIP přepínačů a 8 pull-up rezistorů. Každý přepínač je přímo připojen na hlavní konektor. [6]



Obrázek 5 - Mini modul 8 DIP s přepínači

6.4 Mini modul SMD Ledek

Tento velmi malý modul, který se přímo zapojuje do konektorů základové desky vývojového systému, je složen z 8 SMD LED diod a 8 sériových rezistorů, připojených na kladné napájecí napětí na konektoru. Diody jsou na desce plošných spojů přehledně označeny čísly od 1 do 8.

Obrázek 6 - Mini modul složený s 8 SMD LED

Další informace o základnové desce a modulech lze získat na internetových stránkách společnosti.

Následující popis modulů je proveden dle. [6]

7 Možnosti komunikace s obvodem FTDI pod operačním systémem Linux

Existují dvě varianty jak komunikovat s obvodem FTDI 245MB pod operačním systémem Linux.

Jednou ze dvou možností je použít moduly[4] jádra, které jsou předpřipraveny pro komunikaci s USB zařízeními. Tyto moduly jsou do jádra implementovány buď dobrovolnými vývojáři pracující na vývoji jádra, nebo jsou poskytovány výrobci hardwaru. Linux od verze jádra 2.6 podporuje různé zařízení USB, i moduly od firmy FTDI modelové řady (FT232). Modul (ftdi_sio) po připojení dokáže komunikovat i s naším testovaným obvodem FT245.

Velikou nevýhodou ovladače v jádře je malá využitelnost obvodu z hlediska implementace funkcí pro komunikaci a práci sním. Další nevýhodou je, že funkce jádra neposkytují takový komfort při vývoji aplikací. Pokud chce programátor využít funkcí jádra, tak musí postupovat od začátku a vyvíjet program i s funkcemi pro inicializaci, nastaveni a samotnou komunikaci s obvodem.

Druhou variantou je možnost stáhnout si originální ovladač pro obvody FT245 přímo ze stránek výrobce. S ovladačem dostaneme k dispozici příklady jak otestovat správné připojení obvodu do operačního systému. Z internetových stránek výrobce [10] je možné si stáhnout důležité manuály pro programování API rozhraní. Toto rozhraní nám usnadní samotný vývoj a programování, vzhledem k předpřipraveným funkcím, které můžeme použít.

Díky tomu, že použijeme ovladače od společnosti FTDI získáme lepší pozici při řešení problému, a můžeme snáze využít zákaznické podpory. Další výhodou je, že pro operační systém Windows existuje stejné, nebo velice podobné API rozhraní, díky němuž můžeme získat inspiraci, jak pracovat s API knihovnami.

[4] Modul – jádro je složeno z modulů (ovladačů), tudíž modul je např. ovladač PCI sběrnice, nebo procesu, v našem případe USB zařízení

Z uvedených variant jsem se rozhodl pro druhou možnost, která používá originální ovladač od firmy FTDI. Předpokládám, že mi API rozhranní od výrobce [10] pomůže a usnadní samotný vývoj C++ třídy.

8 Návrh obslužné třídy

Při návrhu obslužné třídy je důležité dbát na požadované vlastnosti, co musí daná třída splňovat a umět. Navrhovaná třída by měla být univerzální pro svou použitelnost v jiných projektech.

8.1 Požadavky na funkce třídy

Než začneme s návrhem, musíme si nejprve ujasnit, co bude třída vykonávat. Třída musí obsahovat vlastnosti a funkce, pomocí kterých budeme moci komunikovat s obvodem.

Požadavky jsou:

- Inicializace obvodu
- Čtení dat příchozích z obvodu
- Zápis dat odeslaných do obvodu
- Reset obvodu
- Ukončení práce s obvodem

8.2 Popis použitých funkcí z API od výrobce

Společnost FTDI nabízí uživateli knihovnu D2xx, která definuje základní komunikační funkce (posílání a příjem bytu, nastavení rozhraní, apod.). Ve staženém souboru ze stránek výrobce [10] nalezneme soubor ftd2xx.h, ze kterého použijeme jednotlivé API funkce pro naši třídu.

8.2.1 Inicializace obvodu

Před použitím integrovaného obvodu je potřeba nejprve provést inicializaci zařízení a nastavit mu např. rychlost čtení, zápisu.

Pro inicializaci a nastavení se využívá funkcí z API a to FT_ListDevice, FT_OpenEx a Set_BaudRate.

Funkce FT_List_Device získá informace o počtu připojených zařízení. Vrací číslo připojeného zařízení a sériové číslo zařízení.

Funkce FT_OpenEX otevírá připojené zařízení a vrací jeho číslo.

Tato funkce FT_SetBaudRate nastaví rychlost přenosu dat v baudech za sekundu.

8.2.2 Zápis do obvodu

Zápis do obvodu je řešen pomocí funkce FT_Write.

8.2.3 Čtení z obvodu

Čtení dat z obvodu je zprostředkováno pomocí funkcí FT_GetStatus a FT_Read. Pokud funkce FT_GetStatus není nulová lze číst data z obvodu pomocí funkce FT_Read.

8.2.4 Reset obvodu

Funkce FT_ResetDevice vyvolá reset a obvodu, používá se pokud se obvod dostane do nějaké nestandardní situace nebo stavu.

8.2.5 Uzavření obvodu

Tato funkce uzavře otevřené zařízení. Pro uzavření obvodu použijeme funkci z API FT_Close.

Následující popis API funkcí je proveden dle. [8]

9 Návrh testovací aplikace

Testovací aplikace bude komunikovat s obvodem FTDI a využívat funkce připojené třídy C++. Tato aplikace musí splňovat následující podmínky:

- Musí používat C++ třídu a její funkce ke komunikaci s obvodem.
- Inicializovat obvod.
- Zapisovat do obvodu přičítáním.
- Provádět reset zařízení.
- Číst do bufferu poslané zprávy z řetězce a zapisovat výsledky na obrazovku.
- Obsahovat vlákno pro čtení.

9.1 Testování funkcí z API rozhranní

V testovací aplikaci budu testovat následující funkce z C++ třídy používající API funkce a to tyto:

ft_init, ft_reset, ft_write, ft_close, ft_read.

Pomocí této testovací funkce ověříme správnou práci obvodu a použití API funkcí.

10 Programová realizace

10.1 Komunikační software uvnitř čipu FTDI 245 BM

Při zápůjčce vývojového kytu mi byl do obvodu nahrán komunikační software. Díky tomuto softwaru budeme moci otestovat funkčnost C++ třídy. Pro vizuální kontrolu správnosti výstupu slouží modul osmi SMD ledek. Modul osmi DIP spínačů slouží k nastavení funkcí programu a k posílání výstupních řetězců (výstup pro čtení).

Vnitřní program čipu a jeho funkce vůči přepínači s DIP tlačítky:

Všechny přepínače OFF

- zobrazí součet uložení hodnoty a přijatých bytů binárně na display 8 SMD LED, funkce přičítání.

Přepínač 1 ON ostatní OFF

- zobrazí binárně na display 8 SMD LED ASCI hodnotu přijatého čísla
- pošleme-li 3, zobrazí na led diodách 51

Přepínač 1 a 2 ON ostatní OFF

- zobrazí binárně na display 8 SMD LED dekadickou hodnotu přijatého čísla
- pošleme-li 3, zobrazí se 3

Všechny přepínače OFF, stisk klávesy 7 do stavu ON a její vrácení do stavu OFF

- vrátí na výstup z obvodu písmeno "a"

Všechny přepínače OFF, stisk klávesy 8 do stavu ON a její vrácení do stavu OFF

- vrátí na výstup z obvodu písmeno "ahoj"

10.2 Zprovoznění ovladače od výrobce FTDI

Jádro jako takové obsahuje velké množství ovladačů od výrobců, nebo ovladačů napsaných vývojáři. Pokud připojíme k počítači námi testovaný obvod, jádro si nahraje moduly pro komunikaci s ním. Jsou to moduly (*ftdi_sio*, *usbserial*).

Pokud chceme použít originální ovladače od společnosti FTDI musíme z kernelu odebrat původní moduly komunikující se sériovými a ftdi obvody. Pro odebrání těchto modulů použijeme utilit *rmmod*[5]. Poté musíme nahrát ovladače (knihovny), komunikující s naším obvodem. Knihovny můžeme stáhnout z internetových stránek výrobce.

10.2.1 Výchozí situace

Použijeme počítač řady PC, který obsahuje alespoň jeden USB konektor. Na počítač nainstalujeme operační systém Linux - Ubuntu 6.06 LTS - Dapper Drake vydaný v roce 2006. Verze jádra Ubuntu 6.06 LTS je 2.6.15-52-386. Jedná se o 32 bitovou verzi systému. Jádro zůstane v původní konfiguraci, jak bylo nainstalováno v průběhu instalace. Do jádra jsme nenahráli knihovny od společnosti FTDI, ani jsme neprováděli odebírání modulů z jádra.

10.2.2 Úprava jádra

Nyní si ukážeme postup jak upravit jádro systému a zprovoznit námi testovaný obvod pomocí originálního ovladače od společnosti FTDI.

Ovladač D2XX vytvoří v systému nový virtuální COM port a veškerá komunikace pak probíhá přes něj. Ovladače D2XX nabízí uživateli DLL knihovnu, která definuje základní komunikační funkce (poslání a příjem bytu, nastavení rozhraní, apod). [7]

[5] rmmod – jednoduchý program odebírající moduly z jádra Linuxu

Tento praktický příklad je prováděn v terminálu s přihlášeným super uživatelem „root“.

Praktický příklad vložení ovladače do systému:

1. Nejdříve je potřeba stáhnout knihovnu ze stránek společnosti. [11]
2. Protože stažená knihovna *libftd2xx0.4.13.tar.gz*. je z důvodu velikosti pro účely stahování zabalená, je nutné ji nejdříve rozbalit.
3. Knihovnu rozbalíme například do adresáře "testFTDI"

   -gunzip libftd2xx0.4.9.tar.gz

   - tar -xvf libftd2xx0.4.9.tar

   Výpis rozbaleného adresáře zařídíme použitím příkazu *ls-la*

```
mik@mik-laptop:~/Desktop/testFTDI$ ls –la
drwxr-xr-x  5 mik mik   4096 2008-07-28 14:40 ..
   drwxr-xr-x  8 mik mik   4096 2008-07-26 18:08 ..
   -rw-r--r--  1 mik mik   1202 2006-11-20 18:22 Config.txt
   -rwxr-xr-x  1 mik mik   3183 2006-11-20 18:22 FAQ.txt
   -rw-r--r--  1 mik mik    110 2006-11-20 18:22 ftd2xx.cfg
   -rwxr-xr-x  1 mik mik  20657 2006-11-20 18:22 ftd2xx.h
   -rw-r--r--  1 mik mik 993280 2008-04-03 21:31 libftd2xx0.4.13.tar
   -rwxr-xr-x  1 mik mik 319466 2006-11-20 18:22 libftd2xx.so.0.4.13
   drwxr-xr-x  2 mik mik   4096 2006-11-20 18:22 lib_table
   -rwxr-xr-x  1 mik mik   5698 2006-11-20 18:22 README.dat
   drwxr-xr-x 12 mik mik   4096 2006-11-20 18:22 sample
   drwxr-xr-x  2 mik mik   4096 2006-11-20 18:22 static_lib
-rwxr-xr-x  1 mik mik   2230 2006-11-20 18:22 WinTypes.h
```

4. Nakopírujeme knihovnu do */usr/local/lib*

   cp libftd2xx.so.0.4.13 /usr/local/lib

5. Přesuneme se do adresáře */usr/local/lib*

   cd /usr/local/lib

6. Vytvoříme potřebný symbolický odkaz

   ln -s libftd2xx.so.0.4.13 libftd2xx.so

7. Přesuneme se do adresáře /usr/lib

   cd /usr/lib

8. Vytvoříme potřebný symbolický odkaz

   ln -s /usr/local/lib/libftd2xx.so.0.4.13 libftd2xx.so

9. Přidáme následující řádek do */etc/fstab*
   none /proc/bus/usb usbdevfs defaults,devmode=0666 0 0
   pokud máme jádro vyšší verze 2.6, použijeme *usbfs* a vložíme následující řádek: *none /proc/bus/usb usbdevfs defaults,mode=0666 0 0*
   soubor uložíme

10. Obnovíme připojení všech zařízení uvedených v /*etc/fstab*
    mount –a

11. Pokud máme nainstalováno jádro novější verze - 2.6 a vyšší musíme provést ještě následující krok a odebrat moduly pro detekci FTDI USB zařízení (*rmmod ftdi_sio*) a (*rmmod usbserial*). Kdybychom tyto moduly z jádra neodebrali, tak by se automaticky použily při zapojení USB konektoru moduly jádra. Tyto moduly by nám znemožnily pracovat plnohodnotně s obvodem pomocí knihovny a programátorské příručky od společnosti FTDI.

10.3 Vývojová platforma

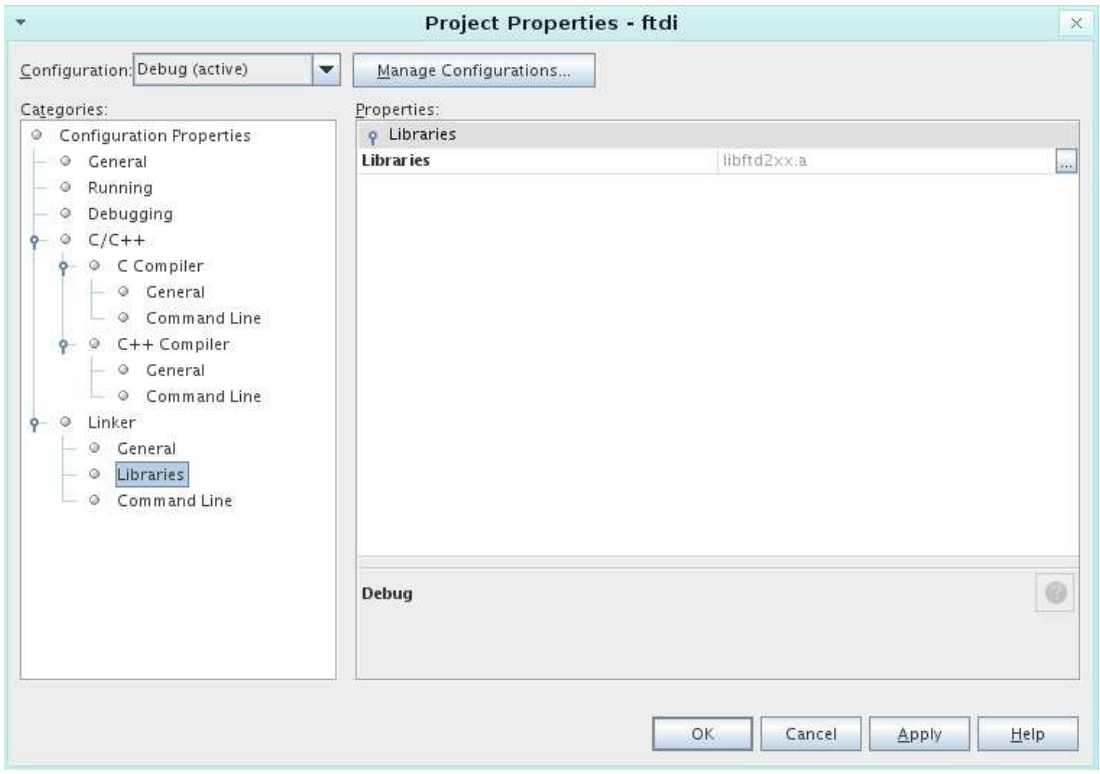
Pro vývoj bylo použito vývojové prostředí NetbeansIDE od společnosti Sun Microsystems (dále jen IDE). Do IDE byly přidány moduly pro vývoj a programování v jazyce C++. Do operačního systému Linux byl nainstalován překladač *gcc*, *g++*, *automake*, *autoconf*, *make*.

V IDE byl vytvořen projekt C++, do něhož byly přidány knihovny pro kompilaci a knihovna pro využití API k obvodu ftdi.

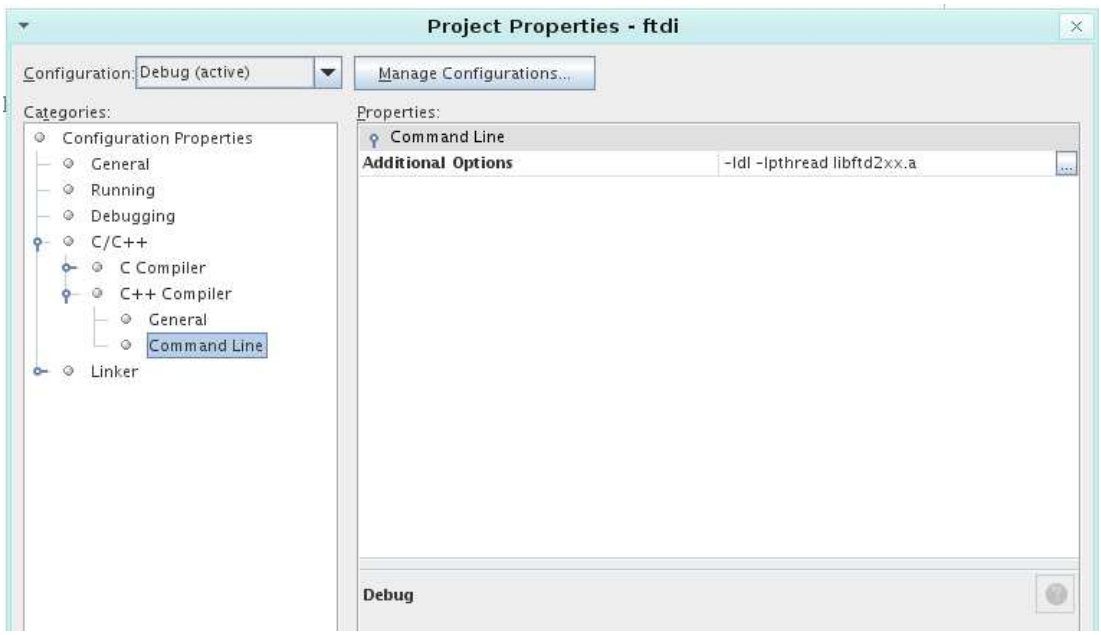
Knihovna pro využití API rozhranní byla použita ze staženého souboru libftd2xx.so.0.4.13 (10.2.2) a byla přejmenována na libftd2xx.a a poté následně přidána do knihoven projektu viz. Obrázek 7 - Vložení knihovny libftd2xx.a do projektu.

Dále bylo nutné nastavit podmínky kompilace pro práci s vlákny, API rozhranním a připojit je do IDE prostředí. Jednalo se o knihovny *ldl*[6], *lpthread*[7], *libftd2xx.a* viz. Obrázek 8 - Vložení funkcí pro překlad ldl, lpthread, libftd2xx.a.

[6] ldl – slouží k načtení dynamických knihoven
[7] lpthread – knihovna pro práci s vlákny



Obrázek 7 - Vložení knihovny libftd2xx.a do projektu



Obrázek 8 - Vložení funkcí pro překlad ldl, lpthread, libftd2xx.a

10.4 Testovací aplikace

Pro účely testování jsem vytvořil aplikaci, která ověří funkčnost třídy pro práci s obvodem FTDI 245. Aplikace se spouští pod super uživatelem (root) v terminálové relaci, kde na obrazovce spouštíme testy jednotlivých funkcí z nabídkového menu vytvořeného testovací aplikací.

Při spuštění aplikace se provede nejprve inicializace obvodu, načte se USB zařízení a nastaví se důležité parametry pro práci se zařízením např. rychlost. Výsledkem správného načtení USB zařízení a tím ověření základní komunikace je zobrazení sériového čísla obvodu upce0001.

10.4.1 Vyhodnocení funkcí aplikace

Při testování se ověřila správná činnost funkcí z C++ třídy a jejich spolupráce s obvodem FTDI 245BM pod operačním systémem Linux.

Po spuštění aplikace se obvod zinicializoval, zobrazilo se sériové číslo

upce001 a také menu viz. Obrázek 9 - Spuštění testovací aplikace

Obrázek 9 - Spuštění testovací aplikace

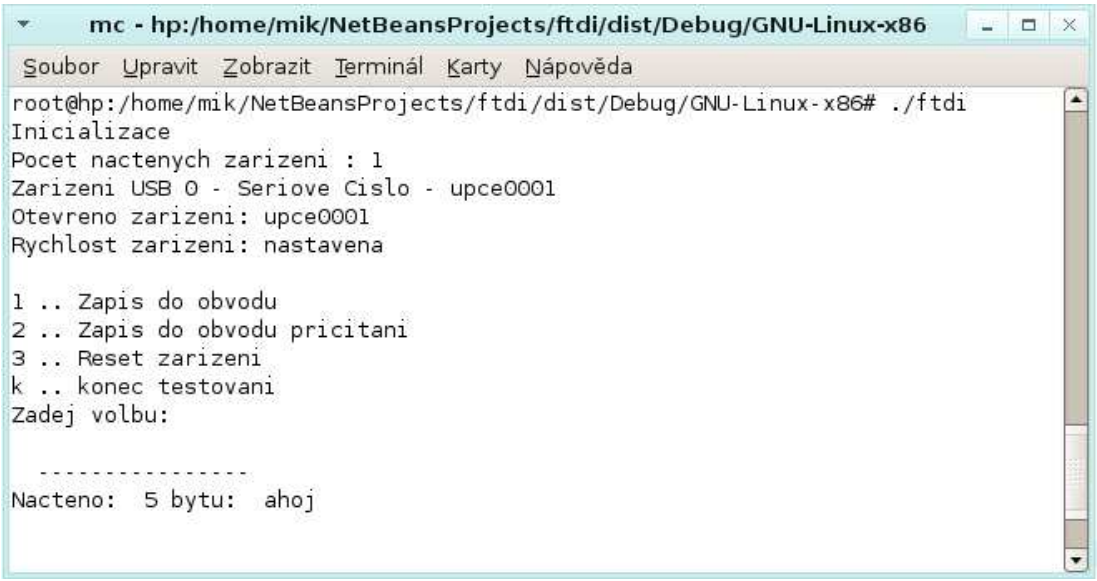
Všechny spínače na modulu DIP byly nastaveny do polohy OFF a do obvodu byly zasílány bity a jejich přijatá hodnota byla přičtena a zobrazena na modulu SMD rozsvícenými LED diodami a počet odeslaných a zobrazených přičtených bitů byl vždy správný.

Spínače na modulu DIP mimo prvního byly nastaveny do polohy OFF a do obvodu byly zaslány znaky z klávesnice a rozsvícené led diody zobrazily binárně jejich ASCI hodnotu (3 -> 51, 2 -> 50). Všechny zaslané hodnoty souhlasily s ASCI tabulkou.

Modul DIP byl nastaven takto: přepínač 1, 2 poloha ON ostatní OFF. Byla zaslána hodnota 2 a rozsvítila se dioda 2. Byla zaslána hodnota 7 a rozsvítila se dioda 1, 2, 4. Dekadická hodnota na led diodách souhlasila se zaslanou hodnotou.

Celý DIP modul byl nastaven do polohy OFF a byl přepnut přepínač 7 do stavu ON a vrácen zpět na OFF. Na terminálu se zobrazilo písmeno „a“.

Všechny přepínače DIP modulu byly nastaveny do polohy OFF a poté byl přepnut přepínač 8 do stavu ON a vrácen zpět na OFF. Na terminálu se zobrazil nápis „ahoj“ viz. Obrázek 10 - Ukázka čtení z obvodu.



Obrázek 10 - Ukázka čtení z obvodu

11 Závěr

Bakalářská práce byla zaměřena na otestování připojení integrovaných obvodů FTDI 245BM pod operačním systémem Linux a bylo použito řešení s použitím originálních ovladačů a API rozhranní od výrobce FTDI obvodů.

Všechny testované funkce pracovaly plnohodnotně a dávaly správné výsledky. Můžeme říci, že práce s obvodem je pod operačním systémem Linux možná a cíle bakalářské práce bylo dosaženo.

Seznam použité literatury

[1] Matoušek, David. USB prakticky s obvody FTDI, 1.díl.
1. vydání. BEN – technická literatura, Praha 2003 ISBN 80-7300-103-9

[2] Informace o mikroprocesorech s rozhraním USB.
Dostupné na WWW: < http://hw.cz/>

[3] Linux začínáme programovat
1. vydání. Computer Press – programování, Praha 2000 ISBN 80-7226-307-2

[4] Informace o mikroprocesorech s rozhraním USB.
Dostupné na WWW: < http://hw.cz/Teorie-a-praxe/Dokumentace/ART327-USB---Universal-Serial-Bus---Popis-rozhrani.html >

[5] Informace o integrovaných obvodech FTDI
dostupné na WWW: < http://noel.feld.cvut.cz >

[6] Informace o vývojových systémech, modulech PK-DESIGN
dostupné na WWW: < http://pk-design.net/HtmlCz/Mainboards.html >

[7] Rozhraní pro připojení k PC přes USB
dostupné na WWW: < http://ksvi.mff.cuni.cz/~krulis/clanky/usb/index.html >

[8] Příručka s API rozhranním pro obvody FTDI 254BM
dostupné na WWW: < http://www.ftdichip.com/Documents/ProgramGuides/D2XXPG34.pdf >

[9] Informace o procesorech, multiprocesingu
dostupné na WWW: < http://kurz.softex.cz/lexikon/vicecpu.html >

[10] Informace o FTDI obvodech, stránka výrobce
dostupné na WWW: < http://www.ftdichip.com >

[11] Knihovny pro obvody FTDI 254BM
dostupné na WWW: < http://www.ftdichip.com/Drivers/D2XX.htm >

[12] Informace o vývojových systémech, modulech PK-DESIGN
dostupné na WWW: < http://www.root.cz/clanky/programovani-pod-linuxem-tema-vlakna/ >

[13] Informace o Linuxových vláknech
dostupné na WWW: < http://www.linux.cz/noviny/1998-809/clanek11.html >

[14] Informace o rozhraní USB
dostupné na WWW: < http://en.wikipedia.org/wiki/Universal_Serial_Bus >

[15] Úvod do operačního systému Linux
dostupné na WWW:
< http://cs.wikibooks.org/wiki/%C3%9Avod_do_OS_Linux >