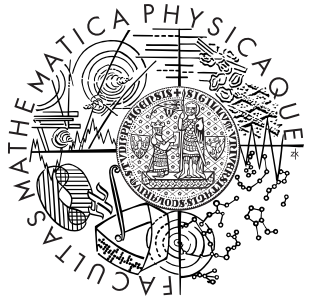
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁRSKA PRÁCA



Kamil Kaščák

Programování ovladačů vnějších zařízení a výhodnocování dat v C++

Ústav částicové a jaderné fyziky

Vedúci bakalárskej práce: RNDr. Peter Kodyš, CSc.

Študijný program: Informatika, Správa počítačových systémů

2009

Rád by som poďakoval RNDr. Petrovi Kodyšovi, CSc. za aktívnu spoluprácu pri tvorbe tejto práce a za zapožičanie hardvéru potrebného k testovaniu vytváraných ovládačov.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce a jej zverejňovaním.

V Prahe dňa 29.7.2009 Kamil Kaščák

Obsah

Názov práce: Programování ovladačů vnějších zařízení a výhodnocování dat v C++
Autor: Kamil Kaščák
Katedra (ústav): Ústav částicové a jaderné fyziky
Vedúci bakalárskej práce: RNDr. Peter Kodyš, CSc.
e-mail vedúceho: peter.kodys@mff.cuni.cz

Abstrakt: V predloženej práci študujeme stav ovládačov externých zariadení v čistej miestnosti v Ústave časticovej a jadrovej fyziky. Hlavná časť práce sa venuje rozboru existujúceho softvéru a tvorbe nového ovládača pre komunikáciu cez RS232 port. Sú skúmané rôzne druhy a spôsoby komunikácie cez sériový port pod operačným systémom Windows aj Linux. Práca poskytuje návrh možného riešenia komunikácie, ktoré spočíva vo vytvorení univerzálnej vrstvy, ktorá bude zabezpečovať komunikáciu medzi užívateľskými programami a externými zariadeniami. Ďalej je poskytnutý popis dôležitých krokov pri implementácii ovládača pod operačným systémom Windows aj Linux. Nasleduje podrobný popis použitia vytvoreného softvéru a popis odlišností pri použití v jednotlivých operačných systémoch. Na záver práca poskytuje popis priebehu testovania ovládača na reálnych zariadeniach a rady k tvorbe podobných softvérových riešení.

Klúčové slová: RS232, sériová komunikácia, ovládač, externé zariadenie

Title: Programming of drivers of external devices and data evaluation in C++
Author: Kamil Kaščák
Department: Institute of Particle and Nuclear Physics
Supervisor: Peter Kodyš, Ph.D.
Supervisor's e-mail address: peter.kodys@mff.cuni.cz

Abstract: In the present work we study current status of drivers for external devices in clean room in Institute of Particle and Nuclear Physics. In this work we analyse existing software and create a new driver for communication through RS232 port. Work studies various possibilities of serial communication in Windows and Linux and creates possible design for driver, which is based on creating universal layer for supplying communication between user programs and external devices. Work describes important steps of implementation in both operating systems. Next part consists of user documentation for created software. This documentation focuses on differences of usage in Windows and Linux. At the end work shows report from testing software with real devices and gives advices, which should programmer follows when creating similar software solution.

Keywords: RS232, serial communication, driver, external device

Kapitola 1

Úvod

Práca vznikla ako potreba Ústavu časticovej a jadrovej fyziky Matematicko-fyzikálnej fakulty Karlovej Univerzity. Viaže sa na laboratória pod názvom čisté miestnosti, kde prebiehajú fyzikálne pokusy ovládané riadiacimi počítačmi. K riadiacim počítačom sa pripájajú externé zariadenia, pomocou ktorých sú pokusy vykonávané. Pre potreby fyzikálnych pokusov je nutné so zariadeniami správne komunikavať.

Motiváciou pre túto prácu boli:

- Úplná absencia softvéru pre zabezpečenie komunikácie s externými zariadeniami.
- Problémové správanie existujúcich softvérových riešení.
- Malá prenositeľnosť existujúcich softvérových riešení pri ovládaní rôznych zariadení.

Cieľom práce bolo zmapovanie situácie so súčasnými ovládačmi pre zariadenia v čistej miestnosti. Ďalším krokom bol výber konkrétneho problému, respektíve komunikačného rozhrania, tak aby vyriešenie tohto problému náročnosťou odpovedalo potrebám bakalárskej práce. Najdôležitejšou časťou práce je samotný návrh riešenia na zabezpečenie komunikácie cez vybrané rozhranie, implementácia tohto návrhu a otestovanie vytvoreného riešenia na reálnych zariadeniach.

Kapitola 2

Popis zariadení a dostupných ovládačov v laboratóriu

2.1 Používané rozhrania

Medzi používané rozhrania na pripájanie externých zariadení v laboratóriu patria : USB, RS232 (tiež známe ako RS232C, RS422, RS485), RJ45, NXI-VXI-2, GPIB (tiež známe ako IEEE-488, HPIB, IMS), NI-OSC. Nasleduje tabuľka, ktorá vymenováva najdôležitejšie zariadenia v laboratóriu spolu s rozhraním, cez ktoré sa pripájajú k počítačom.

Externé zariadenie	rozhranie	počet zariadení
DAQ - SCT	NXIVXI-2	1
DAQ - DEPFET	USB	2
optický atenuátor	RS232	1
polohovací stolček	RS232	5
generátor napätia	GPIB	2
zdroj vysokého napätia	GPIB	2
zdroj nízkeho napätia	RS232	2
osciloskop	GPIB	1
voltmeter	RS232	2
merač optického výkonu	RS232	1

2.2 Stav ovládačov

Súčasný stav ovládačov je z veľkej miery ovplyvňovaný dostupnosťou ovládačov dodávaných výrobcom spolu so zariadením. Pri takýchto zariadeniach problémy s funkčnosťou nenastávajú. K problémom dochádza až v prípade, keď ovládače podporujú iba jeden operačný systém. To vyvoláva potrebu špecifických a často neúplných riešení, ktoré vedú k nespoľahlivosti a problémom s funkčnosťou. Ovládače, ktoré neboli dodané spolu so zariadením bolo potrebné

naprogramovať. Tieto ovládače sú väčšinou neúplné, vykazujú problémy a nečakané správanie pri chode alebo sú príliš zviazané s konkrétnym zariadením. Nasleduje tabuľka popisujúca stav ovládačov pre jednotlivé rozhrania a operačné systémy.

Rozhranie	OS Windows	OS Linux
USB	funkčné	nepostačujúce
RS232	nepostačujúce	nepostačujúce
RJ45	nefunkčné	nefunkčné
NXI-VXI-2	funkčné	nefunkčné
GPIB	funkčné	nefunkčné
NI-OSC	funkčné	nefunkčné

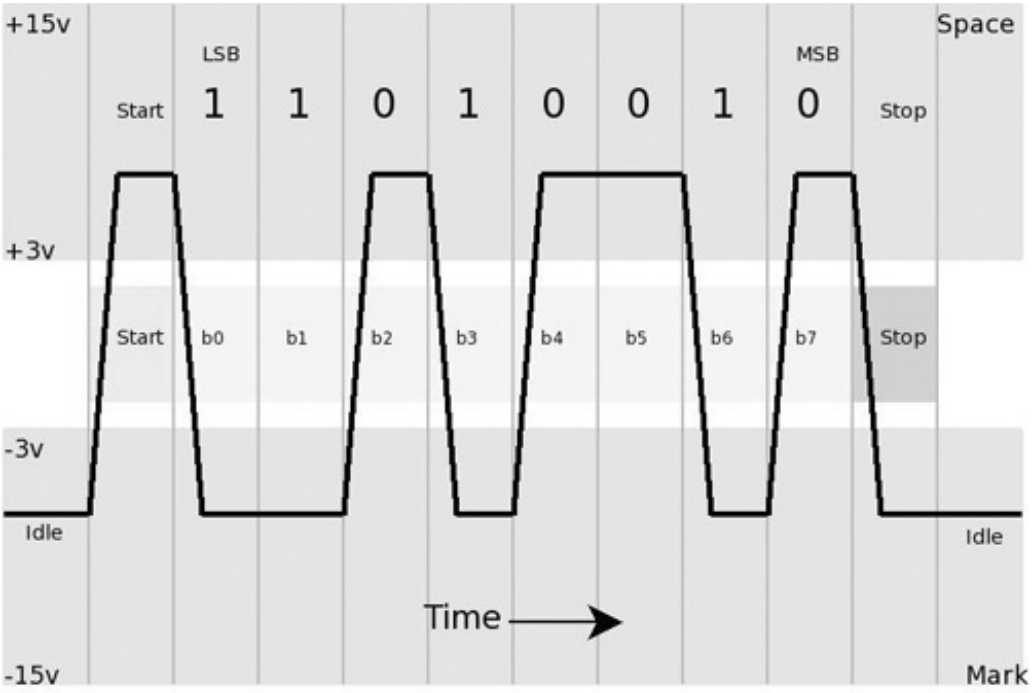
2.3 Rozhranie RS232

Komunikáciu cez rozhranie RS232 (obrázok 1) označované aj ako sériový port alebo sériová linka, ako je uvedené v [5], definuje štandard z roku 1969 pod názvom RS-232C. Existujú aj modifikácie štandardu a to RS-422 a RS-485. Toto rozhranie sa používa na pripájanie a ovládanie mnohých zariadení v skúmanom laboratóriu. Sériových portov je v súčasných počítačoch málo, prípadne úplne chýbajú. Na pridávanie potrebných portov sa používajú USB na RS232 konvertory. Tieto konvertory potrebujú pre svoj chod funkčný ovládač, ktorý zabezpečí pridanie nového sériového portu spôsobom, že program komunikujúci so zariadením cez takto pridaný RS232 port nezaznamenáva žiadny rozdiel v porovnaní s portom zabudovaným v matičnej doske riadiaceho počítača. Tento ovládač je však bežne dodávaný len pre operačný systém Windows. Ovládače pre systém Linux existujú, ale sú poskytované iba niektorými výrobcami týchto konvertorov. Nedostatok konvertorov funkčných pod Linux-om je jeden z dôvodov pre potrebu riadiaceho počítača s operačným systémom Windows, hoci cieľom by bol úplný prechod na systém Linux.

Obrázok 1 (prevzatý z [5]): RS-232 konektor

Pre prenos dát sa používa asynchrónna sériová komunikácia. Poradie prenosu dátových bitov je od najmenej významného bitu (LSB) po bit najvýznamnejší (MSB). Počet datových bitov je voliteľný, obvykle se používá 8 bitov, je však možné nastavenie aj štyroch až siedmych dátových bitov v operačnom systéme Windows, v Linuxe možnosť štyroch dátových bitov chýba. Podľa potrieb pripojeného zariadenia je možné nastavenie rýchlosti spojenia a to od 75 do 128000 bitov za sekundu. Možné rýchlosti komunikácie nie je možné nastaviť na ľubovoľnú hodnotu medzi týmito hraničnými hodnotami. Presné hodnoty pre systém Windows je možné zistiť na [4] a pre systém Linux v [3].

Hoci komunikujúce zariadenia poznajú rýchlosť, akou sa dáta prenášajú, musí prijímač začať prijímať v správny okamih, teda musí prebehnúť synchronizácia. V prípade synchrónnej komunikácie súbežne s dátovým vodičom existuje i synchronizačný vodič, na ktorom vysielač oznamuje príjímaču, kedy poslal dáta. Pri asynchrónnej komunikácii sa synchronizačný vodič nepoužíva, naopak vysielač pošle nejaké údaje po dátovom vodiči, po ich prijatí sa prijímač zosynchronizuje. V prípade RS232 každej sekvencii dátových bitov predchádza jeden štart bit, ktorým sa logická hodnota na linke prepne (pôvodne v pokojovom stave) do opačného stavu. Po dátových bitoch nasleduje paritný bit a za ním jeden alebo viac stop bitov, počas ktorých je linka opäť v pokojovom stave. Počet stop bitov a parita sa môže meniť podľa potrieb zariadenia a možností v jednotlivých operačných systémoch. Príklad prenosu demonštruje obrázok 2.



Obrázok 2 (prevzatý z [5]):Prenos znaku K (ASCII kód 75, binárne 01001011) bez parity a s jedným stop bitom

V súčasnej dobe sa od použitia RS232 portov v osobných počítačoch definitívne upustilo. Tento port je nahradený modernejším a výkonnejším USB portom. V niektorých oblastiach však používanie tohto portu stále prebieha, hlavne z dôvodu jeho jednoduchosti.

Kapitola 3

Výber problému a návrh riešenia

3.1 Popis problému

Pri používaní a ovládaní externých zariadení v laboratóriu narážajú užívatelia na niekoľko problémov. Tieto problémy sa spájajú hlavne s úplnou absenciou potrebného softvéru na ovládanie niektorých zariadení alebo s absenciou kvalitného softwarového riešenia, a tak sú nútený používať rôzne čiastočne funkčné riešenia plné záplat a iných nesystémových postupov.

Po zmapovaní súčasnej situácie v úvodnej časti práce bola ako vážny problém označená komunikácia cez rozhranie RS232.

V operačnom systéme Linux funkčné riešenie pre komunikáciu cez sériový port existuje, avšak je plne zviazané s konkrétnym zariadením, a preto neumožňuje použitie tohto softvéru pre komunikáciu s iným zariadením.

V operačnom systéme Windows sú dostupné softvérové riešenia pre komunikáciu platené, málo univerzálne, alebo im chýba možnosť volania z príkazového riadku. So softvérom používaným v súčasnosti sa spájajú problémy v podobe neželaného správania. Tieto riešenia slúžia ako ukážky možných spôsobov komunikácie a na načerpanie námetov pri tvorbe vlastného riešenia.

Práca sa teda plne venuje vyriešeniu týchto problémov a vytvoreniu všeobecnej knižnice na komunikáciu cez RS232 port pod oboma operačnými systémami.

3.2 Návrh riešenia

Prvým krokom pri návrhu riešenia bolo zmapovanie súčasného softvéru na komunikáciu cez RS232 port a snaha o identifikovanie jeho chýb a nedostatkov.

Finálny program pre operačný systém Windows sme sa rozhodoli postaviť na základoch existujúceho softvéru tak, aby už existujúce nadstavbové programy nebolo potrebné zásadne meniť. Najprv išlo iba o doplnenie kontrolných subrutín, avšak po konzultáciach došlo aj k zásadnejšej zmene funkčnosti ovládača. Zmena spočíva v tom, že pôvodný ovládač potreboval pri čítaní z portu počet bytov, ktoré ma prečítať, nový ovládač by tento parameter už nevyžadoval. Táto zmena prináša ráznejší zásah do existujúceho softvéru a vedie k potrebe zásadných zmien v nadstavbových programoch.

Pre systém Linux je snaha o zachovanie rovnakej funkčnosti tak, aby programy, ktoré daný ovládač používajú, museli byť pri prechode na tento systém menené čo najmenej. K zásahom v nadstavbových programoch musí prísť z dôvodu odlišnosti nastavovania parametrov spojenia v jednotlivých operačných systémoch.

Návrh počíta s tým, že program bude postavený tak, aby sa neviazal na jedno konkrétne zariadenie, ale aby umožňoval ovládanie viacerých zariadení a rôzne nastavenie parametrov spojenia podľa potrieb jednotlivých zariadení. Cieľom je vytvorenie univerzálnej vrstvy slúžiacej na zabezpečenie komunikácie medzi užívateľskými programami a externými zariadeniami.

Riešenie bude obsahovať dva samostatné programy pre Windows a Linux. Implementácia týchto programov sa bude zásadne líšiť, avšak z užívateľského hľadiska budú poskytovať rovnaké služby. Podstatnejší rozdiel z pohľadu užívateľa bude iba v nastavovaní parametrov spojenia.

Kapitola 4

Riešenie

4.1 Analýza existujúceho softvéru

Z pohľadu užívateľa sa s existujúcim softvérom používaným na komunikáciu s externými zariadeniami cez RS232 port pod systémom Windows spájali určité problémy. Hlavným problémom bola neprítomnosť kontroly vykonania alebo nevykonania príkazu. To je z pohľadu automatizácie úplne neprípustné správanie. Ďalším neželaným javom bolo občasné zamrznutie programu. Veľkým problémom z pohľadu používania služieb ovládača bola potreba dopočitávania, a v mnohých prípadoch dokonca hádania počtu bytov, ktoré majú byť prečítané. Ak bol odhad nadhodnotený, mohlo dôjsť k zastaveniu práce ovládača až do uplynutia časových limitov. To vedie k zbytočnému spomaleniu práce nadstavbových programov. V prípade, že odhad bol menší ako skutočný počet bytov odpovede zariadenia, dochádzalo k problémom s nasledujúcim čítaním, pretože neprečítané byty ostávali v bufferoch. To si vyžadovalo rôzne záplaty na dočítanie alebo zbavenie sa týchto neočakávaných bytov a tieto záplaty bolo potrebné meniť v závislosti na zariadení.

Z programátorského hľadiska bol ovládač napísaný mierne neprehľadne. Úplne bolo ignorované správne nastavovanie návratových hodnôt v prípade chyby v chode metódy. Chýba kontrola vykonania alebo nevykonania mnohých kľúčových príkazov. Pri použití dôležitých knižničných funkcií sa neošetrujú všetky možné ukončenia a mnohé situácie, ktoré ovládač vyhodnotí ako úspešne znamenajú pre nadstavbový program chybu a potrebné zastavenie chodu neprichádza. To môže viesť k tomu, že chyba sa neprejaví na mieste, na ktorom nastala, a to si vynucuje záplaty v nadstavbovom programe. To je pre užívateľa ovládača veľmi nepraktická situácia. Za hlavný problém však považujeme to, že program je napísaný takým spôsobom, pri ktorom mnohé jeho vlastnosti vôbec nie sú používané. Existujúci program implementuje prekrývajúcu sa (asynchrónnu) komunikáciu, pričom jej potenciál a výhody sú využívané minimálne. Takto navrhnutá komunikácia spolu s chýbajúcou kontrolou toho, ako sa môže komunikácia správať a do akých stavov sa môže dostať, vedie k neželanému správaniu ovládača a následne celého programu, ktorý tento ovládač

používa.

Hlavnou chybou softvéru v systéme Linux je jeho naviazanie na jedno konkrétne zariadenie a tým pádom absencia všeobecnej vrstvy, ktorá by sa dala v tomto operačnom systéme použiť aj na komunikáciu s inými zariadeniami. Z pohľadu tejto práce táto aplikácia slúži len na načerpanie myšlienok, a ako ukážka funkčného hoci málo všeobecného riešenia v operačnom systéme Linux.

4.2 Popis riešenia

4.2.1 Ovládač pre operačný systém Windows

Prvým krokom pri tvorbe tohto ovládača bolo preskúmanie možných spôsobov a dostupných vlastností komunikácie. Podstatou skúmania bol výber medzi synchrónnou a asynchrónnou komunikáciou, ktorých presný popis je možné nájsť v [1]. Pôvodný ovládač fungoval asynchrónne ale vo svojej podstate asynchrónnosť nepotreboval. Potreba asynchronity sa však prejavuje pri plnení základnej úlohy ovládača, ktorá spočíva v tom, že zariadeniu sa vyšle príkaz a prečíta sa odpoveď. Pri synchrónnej komunikácii by nastal problém v tom, že ak by program najprv z portu čítal, znamenalo by to zablokovanie programu a potrebný príkaz by sa nikdy neposlal a očakávaná odpoveď by sa nikdy neprijala. V opačnom prípade by ovládač najprv poslal príkaz a až potom by očakával odpoveď. Pri testovaní tohto spôsobu komunikácie s reálnym zariadením však dochádzalo k situácii, že v momente, keď sa program snažil načítať odpoveď, žiadne dáta nedorazili a program sa zablokoval. Toto bol dôvod pre voľbu asynchrónneho spôsobu komunikácie. Ďalším dôležitým aspektom skúmania okrem základného spôsobu komunikácie boli ostatné vlastnosti spojenia. Knižnica *Windows.h* poskytuje na správu vlastností, akými sú rýchlosť spojenia, počet dátových bitov a podobne, štruktúru s názvom DBC = Device-Control Block, ktorá ako svoje dátové položky udržiava potrebné vlastnosti. Podobná štruktúra existuje aj pre správu časových limitov čítania a zápisu.

Komunikácia s externým zariadením cez rozhranie RS232 ma svoje odzrkadlenie v práci so súbormi. Vykonáva sa pomocou rovnakých knižničných funkcií s potrebne odlišnými parametrami. Medzi tieto základné funkcie patria *CreateFile* , *ReadFile* a *WriteFile*, ktorých popis funkčnosti a použitia je možné nájsť na [4]. Veľmi dôležité je preskúmanie možného správania sa týchto funkcií a funkcií na nich priamo závislých. Ako príklad dôležitosti tohto skúmania uvediem funkciu *WriteFile*. Táto funkcia úspech alebo neúspech vykonania signalizuje pomocou návratovej hodnoty, avšak môže nastať situácia, keď funkcia signalizuje úspech, ale bola odoslaná iba časť dát určených na poslanie. Táto situácia by sa iba z návratovej hodnoty identifikovať nedala a chyba by nebola identifikovaná včas a mohla by priniesť neočakávané správanie nadstavbového programu.

Program je implementovaný v jazyku C++ vo vývojovom prostredí Microsoft Visual Studio 2008. Základ programu tvorí trieda TRS232 a niekoľko pomocných funkcií. Úplný obsah hlavičkového súboru sa nachádza v prílohe.

Základné metódy triedy RS232:

- metóda na otvorenie spojenia, nastavenie jeho parametrov a naplnenie potrebných dátových položiek triedy ... *Init(...)*
- metóda na zatvorenie spojenia ... *Close(...)*
- metóda na zápis reťazca do portu ... *Write(...)*
- metóda na prečítanie reťazca z portu na základe reťazca, ktorý sa vyšle ... *Read(...)*

Metóda Init má za úlohu otvorenie spojenia a nastavenie vlastností spojenia, ktoré prišli ako vstupné parametre. Z dôvodu, aby mohlo byť spojenie čo najvšeobecnejšie, boli ako vstupné parametre vybrané: rýchlosť spojenia(baudrate), počet dátových bitov, druh parity, počet stop bitov a samozrejme číslo COM portu, na ktorý sa má spojenie nadviazať. Ďalšie dva parametre slúžia pre potreby ukončenia čítania a sú to samotný ukončovací reťazec a jeho dĺžka. Spojenie sa nastaví pomocou štruktúr DCB a COMMTIMEOUTS. Pri všetkých krokoch sa kontrolujú návratové hodnoty a v prípade neúspechu sa vydá chybové hlásenie a ďalšie vykonávanie príkazov tejto metódy je okamžite zastavené.

Úlohou metódy Write je podľa očakávania zápis reťazca do otvoreného portu. Vstupnými parametrami sú reťazec znakov, ktorý sa má zapísať a jeho dĺžka. V metóde sa vytvoria potrebné štruktúry nato, aby zápis fungoval asynchrónne. Kontrolujú sa návratové hodnoty a úplnosť zápisu, to znamená, či boli odoslané všetky byty určené na poslanie.

Metóda Read potrebuje rovnako ako Write vytvoriť dátové štruktúry potrebné k asynchrónnej komunikácii. Úlohou tejto metódy je načítanie odpovede zariadenia, ktorá vzniká na základe príkazu, ktorý táto metóda pošle. Na poslanie príkazu sa používa metóda Write. Po zapísaní inštrukcie pre zariadenie program číta odpoveď zariadenia. Čítanie prebieha po jednom znaku, a napriek tomu, že komunikácia prebieha asynchrónne v prípade, že žiaden znak nebol ihneď načítaný program sa zablokuje a každú sekundu skontroluje či očakávaný byt už dorazil, a to až do chvíle kým neuplynie stanovený časový limit. Po uplynutí tohto limitu ovládač vyhlási chybu a skončí svoj chod. Čítanie po jednom znaku prebieha až do momentu kým sa nedetekuje ukončovací reťazec, ktorý bol ovládaču dodaný pri inicializácii spojenia. Návratová hodnota signalizuje úspešnosť vykonania.

4.2.2 Ovládač pre operačný systém Linux

Ovládač pre systém Linux je implementovaný nezávisle na ovládači pre operačný systém Windows. Jeho snahou je poskytnutie rovnakých služieb pre užívateľa ako ovládač pre Windows. S programovacieho hľadiska došlo oproti Windowsovej verzii k zásadným zmenám. Prístup k portu má síce rovnakú myšlienku, a tou je rovnaký prístup ako k súboru, avšak fungovanie dôležitých knižničných funkcií sa zásadne líši. Hlavná zmena nastala pri nastavovaní parametrov spojenia. Rovnako ako v systéme Windows tu existuje celistvejšia štruktúra na udržiavanie týchto parametrov, ktoré je potrebné nastavovať pomocou bitových operácií, ako je uvedené v [2]. Táto mierna nešikovnosť nastavovania viedla k potrebe vybudovania nadstavby pri nastavovaní niektorých parametrov spojenia. Príkladom takejto nadstavby je nastavenie parity. Užívateľ ma na výber tri druhy parity, a keďže typ parity chce užívateľ dodať ako jeden parameter rovnako ako v ovládači pre systém Windows, nadstavba preto vyzerá takto:

```
//Set parity option
switch(l_Parity) {
   case 0: //set no parity
      options.c_cflag &= ~PARENB;
      options.c_cflag &= ~PARODD;
      break;
   case 1: // set odd parity
      options.c_cflag |= PARENB;
      options.c_cflag |= PARODD;
      options.c_iflag |= (INPCK | ISTRIP);
      break;
   case 2: //set even parity
      options.c_cflag |= PARENB;
      options.c_cflag &= ~PARODD;
      options.c_iflag |= (INPCK | ISTRIP);
      break;
   default: printf("Wrong parity option.\n");
      return 0;
}
```

V systéme Linux ja na výber viac spôsobov komunikácie. Naskytuje sa možnosť použitia synchrónnej a asynchrónnej komunikácie. Z rovnakých dôvodov ako v systéme Windows sme sa rozhodli použiť asynchrónnu komunikáciu. V Linuxe pribúda možnosť výberu medzi kanonickým a nekanonickým spracovaním vstupu, ako je uvedené v [2]. Pri kanonickom spracovaní načítavanie funguje riadkovo. to znamená, že funkcia *Read* vráti iba celý riadok ukončený znakom koniec riadka alebo koniec súboru. Tento spôsob komunikácie je bežný

pre terminály, nedá sa však predpokladať, že každé zariadenie musí fungovať a odpovedať takýmto spôsobom. Preto sa ako rozumný krok javí použitie nekanonickej komunikácie. Pri tomto spôsobe spracovávania vstupu funkcia *Read* skončí na základe nastavenia dvoch čiastkových parametrov spojenia. Tými sú: *c_cc[VTIME]* nastavuje znakový časovač *c_cc[VMIN]* nastavuje minimálny počet znakov, ktoré je potrebné prijať, kým funkcia skončí. Ak je vo vyrovnávacej pamäti viac znakov ako MIN, tieto znaky sa prečítajú. Horným obmedzením je jeden z parametrov funkcie *Read*. Rôznou kombináciou nastavenia týchto parametrov je podľa [2] možné dosiahnuť nasledujúce druhy správania funkcie *Read*:

- MIN $>$ 0 a TIME $=$ 0 : MIN je počet prečítaných znakov, kým čítanie skončí, časovač sa nepoužije.
- MIN $=$ 0 a TIME $>$ 0 : TIME slúži ako časovač, načitávanie skončí, ak sa prečíta jeden znak alebo kým neuplynie čas nastavený v časovači. V takom prípade nie sú načítané žiadne znaky.
- MIN $>$ 0 a TIME $>$ 0 : TIME slúži ako medziznakový časovač. Načitávanie skončí, ak je počet prečítaných znakov aspoň MIN alebo čas medzi prijatím dvoch znakov presiahne hodnotu v časovači. Časovač sa aktivuje až po prečítaní prvého znaku a po prečítaní každého ďalšieho znaku sa vynuluje.
- MIN $=$ 0 a TIME $=$ 0 : V tomto prípade čítanie skončí ihneď a vrátia sa znaky, ktoré boli v okamihu spustenia prítomné vo vstupnej vyrovnávacej pamäti.

Načítavanie sme sa rozhodli realizovať rovnakým spôsobom ako pod systémom Windows, to znamená, že čítanie prebieha po jednom znaku až kým sa nedetekuje ukončovací reťazec, hodnota parametru MIN musí byť tým pádom nutne nulová. Keďže odpoveď zariadení po odoslaní niektorých príkazov sa môže oneskoriť, voľba teda padne na nastavenie 2 zo zoznamu, kde hodnota TIME slúži ako časový limit pre čítanie. Ak tento čas ubehne a nenačíta sa žiaden znak, ovládač vyhlási chybu a skončí vykonávanie metódy.

Ostatné časti ovládača sa od verzie pre operačný systém Windows zásadne nelíšia. Rozdiely sú zväčša iba v názvoch knižničných funkcií a ich parametroch.

4.3 Použitie vytvoreného programu

Užívateľom vytvorených ovládačov je programátor, ktorý vytvára programy a makrá na prácu s externými zariadeniami, ktoré sa pripájajú pomocou RS232 portu. Vytvorené ovládače zabezpečujú reálnu komunikáciu a odľahčujú užívateľa od nepríjemností spojených s komunikáciou.

Prvým krokom k používaniu je potreba vytvorenia inštancie triedy *TRS232*. Pomocou tejto inštancie môže užívateľ využívať služby poskytované ovládačom.

Poskytovanými službami sú: testovanie pripravenosti ovládača, otvorenie portu, zápis do portu, čítanie z portu na základe príkazu a zatvorenie portu. Služby v jednotlivých operačných systémoch sa takmer nelíšia. Jedinou odlišnosťou, na ktorú užívateľ narazí, sú parametre pri otváraní portu. Ich počet ani zmysel sa síce nemenia, avšak pri zachovaní rovnakých vlastností spojenia je potrebná zmena ich hodnôt. Užívateľ by mal dbať na dôslednú kontrolu návratových hodnôt jednotlivých funkcií a zvoliť správny prístup k situácii, ktorá nastala. Sám ale musí rozhodnúť či ďalším krokom bude opätovný pokus o vykonanie kroku, pri ktorom nastala chyba, alebo ukončenie chodu celého programu.

4.3.1 Nadviazanie a ukončenie spojenia

Aby mohol užívateľ čítať a zapisovať do portu je potrebné nadviazať spojenie. To sa vykoná pomocou metódy:

*int Init(int COM_ID, int BaudRate, int ByteSize, int Parity, int StopBits, char * EndString, int EndStringLength);*

ktorá otvorí port a nastaví parametre spojenia podľa vstupných parametrov. Popis vstupných parametrov v operačnom systéme Windows:

- COM_ID : číslo COM portu, ku ktorému je cieľové zariadenie pripojené.
- BaudRate : rýchlosť spojenia v bitoch za sekundu. Možné hodnoty majú tvar CBR_*, kde * reprezentuje kladné celé číslo. Napríklad CBR_9600 CBR_14400.
- ByteSize : počet dátových bitov.
- Parity : určenie paritnej schémy. Napríklad NOPARITY = žiadna parita, ODDPARITY = nepárna parita, EVENPARITY = párna parita.
- StopBits : nastavenie počtu stopbitov. Možné hodnoty sú: ONESTOPBIT = jeden stopbit, ONE5STOPBITS jeden a pol stopbitu, TWOSTOPBITS = dva stopbity.
- EndString : reťazec, pri ktorého detekovaní sa zastaví čítanie. Mal by to byť reťazec, ktorý ukončuje každú odpoveď zariadenia.
- EndStringLength : dĺžka ukončovacieho reťazca. Na prvý pohľad zbytočný parameter, ktorý získava opodstatnenie v prípade, že ukončovací reťazec má obsahovať reťazcovú ukončovaciu nulu.

Presný popis možných parametrov je možné nájsť na [4].

Popis vstupných parametrov v operačnom systéme Linux:

- COM_ID : číslo COM portu, ku ktorému je cieľové zariadenie pripojené. Dosadzuje sa do reťazca /dev/ttyS(COM_ID - 1).
- BaudRate : rýchlosť spojenia v bitoch za sekundu. Možné hodnoty majú tvar B*, kde * reprezentuje kladné celé číslo. Napríklad B9600 B38400.
- ByteSize : počet dátových bitov v tvare CS5 CS6 CS7 CS8.
- Parity : určenie paritnej schémy. Hodnota 0 znamená žiadnu paritu, hodnota 1 nepárnu paritu a hodnota 2 párnu paritu. Iné hodnoty sú považované za chybné a spojenie sa ukončí.
- StopBits : nastavenie počtu stopbitov. Hodnota 1 znamená jeden stopbit, hodnota 2 dva stopbity. V prípade inej hodnoty sa program zachová rovnako ako pri zlom parametri parity.
- EndString a EndStringLength : význam a použitie týchto parametrov sa oproti systému Windows nemenia.

Presný popis parametrov BaudRate a ByteSize je možné nájsť v [3].

V prípade úspešného otvorenia portu a nastavenia parametrov metóda vracia hodnotu 1. V prípade akejkoľvek chyby, či pri otváraní portu alebo pri zmene nastavení, ovládač vypíše chybné hlásenie s popisom, pri ktorom kroku nastala chyba, nastaví návratovú hodnotu na 0 a okamžite skončí.

Na ukončenie spojenia je potrebné použiť metódu:
void Close(void);

4.3.2 Čítanie a zápis

Ak chce užívateľ zapísať reťazec do portu použije metódu:

*int Write(char *cOut, int iOutSize);*

Použitie sa nemení v závislosti na operačnom systéme.
Popis parametrov:

- cOut : reťazec, ktorý sa odošle zariadeniu.
- iOutSize : dĺžka odosielaného reťazca.

Metóda v prípade úspechu, to znamená, že všetky znaky boli poslané, vráti hodnotu 1 v opačnom prípade vráti 0.

V prípade, že chce užívateľ poslať zariadeniu príkaz a následne si prečítať jeho odpoveď musí použiť metódu:

*int Read(char *cOut, int iOutSize, char *cIn, int *iInSize);*

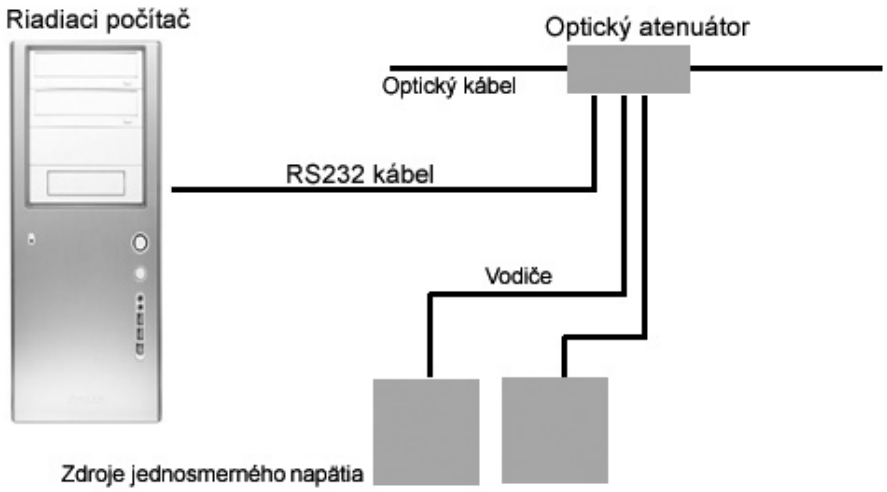
Použitie sa rovnako ako pri metóde *Write* nemení v závislosti na operačnom systéme.
Popis parametrov:

- cOut : reťazec, ktorý sa odošle zariadeniu.
- iOutSize : dĺžka odosielaného reťazca.
- cIn : premenná, do ktorej sa zapíše prečítaný reťazec.
- iInSize : premenná, do ktorej sa zapíše počet znakov, ktoré boli prečítané.

Dôležitým aspektom tejto metódy je ukončovací reťazec, ktorý sa nastavuje pri inicializácii spojenia. V prípade, že tento reťazec nie je medzi prijímanými znakmi nájdený, dôjde k vypršaniu časového limitu a ovládač vyhlási chybu, vypíše chybové hlásenie, ktorého obsahom budú dosiaľ prečítané znaky a skončí s návratovou hodnotou 0. V prípade inej chyby vydá opäť chybové hlásenie, vráti 0 a skončí. Ak je pri čítaní detekovaný ukončovací reťazec, načítavanie sa ukončí a ostatné dáta vo vyrovnávacej pamäti sa zahodia. Metóda v takomto prípade skončí s návratovou hodnotou 1.

4.4 Testovanie vytvoreného programu

Pre potreby testovania sme si vytvorili jednoduchú náhradu užívateľského programu, ktorý zabezpečoval odosielanie príkazov zariadeniam a následné prijímanie odpovedí týchto zariadení. Ovládač bol testovaný na zariadení, ktoré svoj vstup kopíruje na výstup a zariadení s názvom optický atenuátor, ktorého presný názov znie Motor-Driven DD-100-MC In-Line Optical Attenuator (výrobca OZ Optics). Testovanie prebiehalo podľa schémy na obrázku 3.



Obrázok 3: Schéma zapojenia pri testovaní

Komunikácia bola testovaná pod operačným systémom Windows XP na dvoch rôznych počítačoch, v jednom z nich boli zariadenia pripájané priamo k portu na matičnej doske, v druhom boli zariadenia pripájané pomocou USB na RS232 konvertoru. Na oboch počítačoch vykazoval softvér rovnaké a uspokojivé správanie. Softvér pre Linux bol testovaný na distribúcii Ubuntu, kde pri komunikácii nenastávali problémy. Na ďalšom počítači s Linuxovou distribúciou Debian sa komunikáciu nepodarilo sfunkčniť z dôvodu neštandardného alebo chybného mapovania sériového portu. Tento problém bude riešený so správcom daného počítača hardvérovými zmenami.

Kapitola 5

Odporúčania a záver

5.1 Odporúčania pri riešení obdobných softvérových problémov

Pri riešení podobných problémov odporúčame nasledovanie týchto rád:

- Čo najpresnejšia špecifikácia problému, určenie cieľov a požiadaviek na finálny softvér.
- Zmapovanie všetkých možností riešenia. V prípade riešenia komunikácie preverenie všetkých dostupných spôsobov, zistenie výhod a nevýhod jednotlivých spôsobov komunikácie a výber toho najvhodnejšieho pre problém, ktorý chceme vyriešiť.
- Reálne otestovanie čiastkových problémov z dôvodu určenia možnej nekorešpondencie teoretického a praktického riešenia alebo prípadného nepochopenia funkčnosti daného riešenia. Pred návrhom štruktúry programu je vhodné zistiť, čo sa dá použiť, a hlavne, ako to reálne funguje.
- Samotný návrh štruktúry programu a presné určenie úloh jednotlivých častí tohto riešenia.
- V prípade použitia externých knižníc je potrebné presné zmapovanie správania sa knižničných funkcií, ich parametrov a možných návratových hodnôt.
- Pri realizácii použitie: dôslednej kontroly návratových hodnôt, zachytávania nečakaných situácii a včasného ukončenia chodu programu pri výskyte takýchto situácii
- Dôsledné testovanie bežných, hraničných, a aj neočakávaných situácii.
- Vytvorenie kvalitnej užívateľskej dokumentácie.

5.2 Záver

Práca na týchto ovládačoch mi umožnila bližšie preskúmanie a spoznanie vlastností sériovej komunikácie na operačnom systéme Windows aj Linux. Práca mi poskytla skúsenosti s komunikáciou a ovládaním reálnych externých zariadení.

Cieľom práce bolo zmapovanie situácie ovládačov externých zariadení v čistej miestnosti v Ústave časticovej a jadrovej fyziky a voľba vhodného problému na vyriešenie. Voľba padla na vytvorenie ovládača na komunikáciu cez RS232 port. Práca popisuje návrh riešenia, jeho implementáciu a návod na použitie vytvoreného softvéru pre operačné systémy Windows a Linux. Práca ďalej obsahuje popis priebehu testovania a odporúčania pre riešenie podobných softvérových problémov.

Ďalším krokom bude reálne nasadenie softvéru jeho užívateľmi a použitie na väčšom množstve zariadení. Pri tomto použití môže vzniknúť potreba opravy prípadných, testovaním neodhalených chýb, alebo môže nastať potreba väčšej všeobecnosti ovládačov. V takomto prípade dôjde k oprave alebo zmene vytvorených ovládačov podľa potrieb užívateľov.

Ciele práce sú týmto naplnené a do budúcna ostáva otvorená možnosť vytvárania zložitejších ovládačov pre rozhrania ako USB, RJ45, alebo GPIB pre operačné systémy Windows aj Linux. Ďalšou možnosťou ja úprava súčasných aplikácii pre prácu s novými ovládačmi. Tieto možnosti poskytujú námety pre ďalšiu spoluprácu v podobe softvérového projektu alebo diplomovej práce.

Literatúra

[1] Allen Denver: *Serial Communications in Win32*, [online, citované 16.7.2009]. Dostupné na http://msdn.microsoft.com/en-us/library/ms810467.aspx

[2] Gary Frerking & Peter Baumann: *Serial Programming HOWTO*, [online, citované 20.7.2009]. Dostupné na http://www.faqs.org/docs/Linux-HOWTO/Serial-Programming-HOWTO.html

[3] Manuálové stránky k termios.h. [online, citované 20.7.2009] Dostupné na http://www.opengroup.org/onlinepubs/007908799/xsh/termios.h.html

[4] Microsoft Developer Network: *MSDN Library*, [online, citované 15.7.2009]. Dostupné na http://msdn.microsoft.com/en-us/library/default.aspx

[5] Wikipedia: *RS-232*, [online, citované 14.7.2009]. Dostupné na http://cs.wikipedia.org/wiki/RS-232

Príloha A

Hlavičkový súbor ovládača pre Windows

```
#define MAX_TRANSPORT 250
#define READ_TIMEOUT 30 // * 1 second
#include <windows.h>
#include <stdio.h>
typedef struct {
   int Status;
   char SendOut[MAX_TRANSPORT];
   int SendOutSize;
} TransportWrite;

void RS232_Write2(TransportWrite *par);

class TRS232 {
   private:
      //status variables
      int IsInit;
      int COM_ID;
      char EndString[MAX_TRANSPORT];
      int EndString_length;
   public:
      TRS232(){} //constructor
      //test driver dll
      int Test(void);
      //Open and initialize COM serial port
      int Init(int l_COM_ID, int l_BaudRate, int l_ByteSize,
               int l_Parity, int l_StopBits,
               char * l_EndString, int l_EndStringLength);
      //read data based on request string
      int Read(char *cOut, int iOutSize, char *cIn, int *iInSize);
      //write data
      int Write(char *cOut, int iOutSize);
      //close port
      void Close(void);
};
```

Príloha B

Hlavičkový súbor ovládača pre Linux

```
#define MAX_TRANSPORT   250
#define READ_TIMEOUT    2000
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>
#include <sys/signal.h>
#include <sys/types.h>
class TRS232 {
   private:
      int COM_ID;
      char EndString[MAX_TRANSPORT];
      int EndString_length;

   public:
      TRS232(){} //constructor
      //test driver
      int Test(void);
      //Open and initialize COM serial port
      int Init(int l_COM_ID, int l_BaudRate, int l_ByteSize,
               int l_Parity, int l_StopBits,
               char * l_EndString, int l_EndStringLength);
      //read data based on request string
      int Read(char *cOut, int iOutSize, char *cIn, int *iInSize);
      //write data
      int Write(char *cOut, int iOutSize);
      //close port
      void Close(void);
};
```

Príloha C

Obsah priloženého CD-ROM

bc.pdf

Elektronická verzia práce.

RS232_Win_sources

Zdojové kódy ovládača pre Windows.

RS232_Lin_sources

Zdojové kódy ovládača pre Linux.

RS232_Win

Projekt MS Visual Studia obsahujúci ovládač spolu s testovacím programom pre optický atenuátor určený pre operačný systém Windows.

RS232_Lin

Projekt NetBeans obsahujúci ovládač spolu s testovacím programom pre optický atenuátor určený pre operačný systém Linux.