UNIVERZITA PARDUBICE
ÚSTAV ELEKTROTECHNIKY A INFORMATIKY

OVLÁDACÍ PROGRAM K DIGITÁLNĚ REGULOVATELNÉMU LABORATORNÍMU ZDROJI

BAKALÁŘSKÁ PRÁCE

2007 **Petr Křivka**

UNIVERZITA PARDUBICE
ÚSTAV ELEKTROTECHNIKY A INFORMATIKY

OVLÁDACÍ PROGRAM K DIGITÁLNĚ REGULOVATELNÉMU LABORATORNÍMU ZDROJI

BAKALÁŘSKÁ PRÁCE

AUTOR PRÁCE: Petr Křivka
VEDOUCÍ PRÁCE: Ing. Martin Hájek

2007

UNIVERSITY OF PARDUBICE
INSTITUTE OF ELECTRICAL ENGINEERING AND INFORMATICS

CONTROL PROGRAM FOR DIGITALLY REGULATED LABORATORY POWER SUPPLY

BACHELOR WORK

AUTHOR: Petr Křivka
SUPERVISOR: Ing. Martin Hájek

2007

Univerzita Pardubice
Ústav elektrotechniky a informatiky

Vysokoškolský ústav: Ústav elektrotechniky a informatiky
Katedra/Ústav: Ústav elektrotechniky a informatiky
Akademický rok: 2006/2007

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Pro: Křivka Petr

Studijní program: Informační technologie

Studijní obor: Informační technologie

Název tématu: Ovládací program k regulovatelnému laboratornímu zdroji

Zásady pro zpracování:

Navrhněte program pro ovládání regulovatelného napájecího zdroje z počítače (operační systém MS Windows). Komunikace bude probíhat po sběrnici RS232. Program by měl mít následující funkce:

- nastavení výstupních parametrů (napětí, proud),
- přenos a zobrazení měřených hodnot do počítače,
- možnost programování parametrů zdroje v čase a současný přenos naměřených hodnot napětí a proudu,
- grafické zobrazení měřených hodnot a jejich export do souboru, vhodného pro další zpracování (např. MS Excel),
- uložení konfigurace programu do souboru a její obnovení.

Součástí práce také bude:

- rozbor principů programování komunikace pro RS232 pod OS Windows,
- rešerše dostupných komponent pro vývojové prostředky firmy Borland založených na knihovně VCL, zapouzdřujících komunikaci po RS232,
- výběr vhodného způsobu řešení s ohledem na vyvíjený program.

Seznam odborné literatury:

Matoušek, D. *Udělejte si z PC v Delphi elektronickou laboratoř a řídící centrum ve Windows 1. díl*. Praha : BEN, 2003. 271 s.
Sedláček, J., Slaba, J. *Delphi v kostce*. Praha : BEN, 2000. 513 s.
Vykopal, J. *Sériové rozhraní v Delphi*. [online]. 20.11.2001. Dostupné z WWW <http://www.builder.cz/serial85.html>.

Rozsah: Přibližně 30 stran.

Vedoucí práce: Ing. Martin Hájek

Vedoucí katedry (ústavu): prof. Ing. Pavel Bezoušek, CSc.

Datum zadání práce: 30.11.2006

Termín odevzdání práce: 17.8.2007

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 prezenčním zpřístupněním své práce v Univerzitní knihovně Univerzity Pardubice.

V Pardubicích dne 15. 8. 2007

Petr Křivka

Poděkování:

Děkuji vedoucímu bakalářské práce, panu Ing. Martinu Hájkovi za odborné vedení, cenné rady a podnětné připomínky při zpracování práce.

Abstrakt

Bakalářská práce se zabývá problematikou komunikace po sériovém rozhraní RS232 v prostředí Borland Delphi pro OS Windows. Popisuje sériové rozhraní, zásady komunikace po tomto rozhraní, principy programování, dostupné komponenty pro vývojové prostředky společnosti Borland. Praktická část práce se zabývá konkrétním řešením komunikace PC s regulovatelným laboratorním zdrojem prostřednictvím RS232. Program umožňuje nastavení výstupních parametrů (napětí, proud), přenos naměřených hodnot do PC a jejich následné zpracování.

Obsah

Úvod 11
1 Napájecí zdroje 12
1.1 Digitálně regulovatelný zdroj 12
2 Sériové rozhraní 15
2.1 Port COM 15
2.2 Synchronní a asynchronní přenos dat 17
2.3 Parita 18
2.4 Přenosová rychlost (baud rate) 19
2.5 Řízení toku dat 20
2.5.1 Hardwarové řízení toku dat 20
2.5.2 Softwarové řízení toku dat 21
2.6 Serializace dat a SDU 22
2.7 Rozhraní RS323C 24
2.7.1 Logické úrovně RS232C 27
2.8 UART – univerzální asynchronní přijímač-vysílač 28
3 Využiti Borland Delphi pro vývoj řídících aplikací 29
3.1 Události v Delphi (events) 29
3.2 Využití událostí v řídící aplikaci 31
3.3 Vytvoření vlastní události 32
4 Přístup k portům v Borland Delphi 34
4.1 Přímý přístup k portům 34
4.1 Komponenty zapouzdřující komunikaci po RS232 34
4.2 TVaComm 35
4.2.1 Instalace 36
4.2.2 Možnosti čtení a zápisu dat 36
4.2.3 Použité metody 37

5 Ovládací program 39
5.1 Popis třídy TRegZdroj 40
5.2 Komunikační protokol 42
5.3 Zpracování příchozích dat pomocí událostí 42
5.4 Popis programu 43
5.5 Nastavení výstupních hodnot v čase 43
6 Závěr 45

Seznam obrázků

1. Blokové schéma zdroje 13
2. Digitálně regulovatelný laboratorní zdroj 14
3. Standardy sériového rozhraní 15
4. Formát asynchronního přenosu RS232C 17
5. Hardwarové řízení toku 20
6. Softwarové řízení toku dat 21
7. SDU 22
8. Převod a vysílání sériových dat 23
9. Napěťové úrovně signálů RS232C 27
10. Sekvenční přístup 29
11. Vyvolání události 30
12. Příjem a obsloužení událostí 30
13. Aplikace událostí v rámci bakalářské práce. 31
14. Grafické znázornění částí programu 39
15. Hierarchie programu 39
16. Popis třídy TRegZdroj 40
17. Hlavní okno řídící aplikace 43
18. Nastavení výstupních hodnot v čase 44

Úvod

Motivací pro bakalářskou práci bylo vytvořit program – řídící aplikaci, která by komunikovala s konkrétním digitálně regulovatelným laboratorním zdrojem prostřednictvím sériového rozhraní RS232, kterým tento zdroj disponuje.

Pomocí této aplikace bude možné nastavovat aktuální výstupní hodnoty zdroje. Pro automatizaci jednoduchých měřících úloh bude implementována možnost nastavení výstupních parametrů zdroje v čase.

Teoretická část práce čtenáře krátce seznámí s napájecími zdroji. Dále se zabývá principy komunikace po sériovém rozhraní, možnostmi návrhu řídících aplikací ve vývojovém prostředí Borland Delphi provozovaném pod operačními systémy Microsoft Windows, popisem principu událostmi řízeného programování a jeho praktickým využitím pro návrh řídících aplikací.

1 Napájecí zdroje

Napájecí zdroje jsou používané v elektrotechnické praxi, v laboratořích, ale i v běžném životě. Existuje mnoho typů napájecích zdrojů. Od jednoduchých lineárních s pevně nastaveným výstupním napětím přes zdroje s regulací, zdroje spínané až po složité zdroje s možností nastavení různých tvarů signálu na výstupu. V případě analogových zdrojů se k regulaci nejčastěji používá potenciometrů, které ovládají výkonové tranzistory.

Mezi složitější patří zdroje s digitální regulací, kde se výstupní hodnoty nastavují pomocí tlačítek nebo klávesnice připojené k mikrořadiči. Vlastní regulace bývá realizována analogově. Pro spojení analogové a digitální části se používají A/D převodníky. Digitálně řízené zdroje obvykle mají rozhraní pro vzdálené ovládání (připojení k PC nebo k řídícímu zařízení).

Dalším typem zdrojů jsou zdroje spínané, které pracují na vyšších frekvencích, než je frekvence na vstupu. Vstupní napětí se vyfiltruje a převede na napětí s vyšší frekvencí. Získaný signál s frekvencí řádově kHz se lépe transformuje na potřebná napětí, lépe filtruje a lépe stabilizuje. Současně ztráty v transformátoru jsou menší. Tento typ zdrojů se nejčastěji používá ve výpočetní technice.

U klasických zdrojů dochází k transformaci na požadované napětí, které se usměrní. Tím vznikne ze střídavého pulzující stejnosměrné napětí. Následuje vyhlazovací filtr, který potlačí zvlnění výstupního stejnosměrného napětí zdroje na potřebnou mez – ta je dána požadavky napájeného zařízení. Následuje stabilizátor, který funguje současně jako vyhlazovací filtr a proto je v některých případech vynechán.

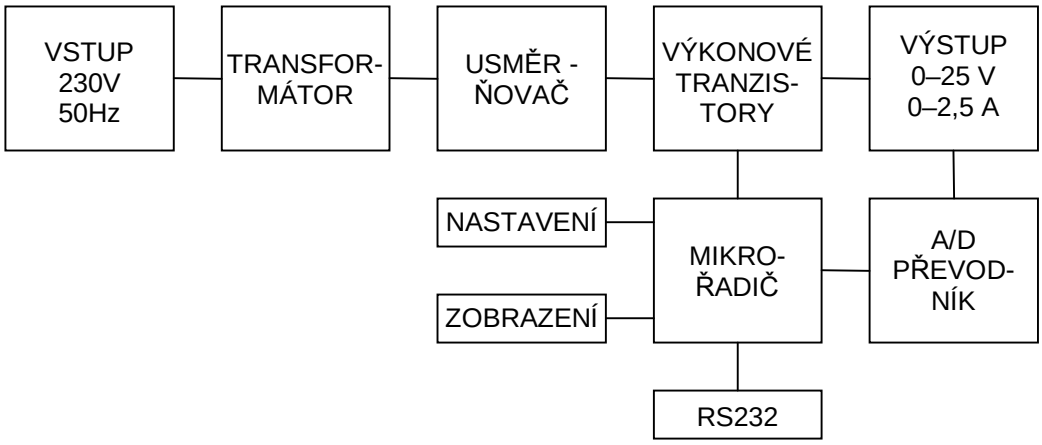
1.1 Digitálně regulovatelný zdroj

Regulovatelný zdroj je základní věcí pro každého, kdo pracuje nebo přijde do styku se slaboproudou elektronikou. Vzhledem k možnosti regulace výstupních hodnot je možné napájet nejrůznější zařízení, využít zdroj pro měření charakteristik elektronických součástek a v neposlední řadě pro oživení nových výrob-

ků, kde se uplatní ochrana proti zkratu nebo nastavení maximálního odběru proudu pro případ chybné konstrukce výrobku.

Zdroj, ke kterému je psán program v rámci této práce je řízen digitálně mikrořadičem PIC. Princip funkce zdroje popisuje blokové schéma na obrázku 1. Výstupní hodnoty se nastavují pomocí dvou čtveřic tlačítek přímo na zdroji. Dvě pro hrubé nastavení a dvě pro jemné nastavení napětí a proudu. Možnosti nastavení výstupních parametrů zdroje jsou od 0–25 V a od 0–2,5 A. Napětí je možno nastavovat po 0,1 V jemně a po 1 V hrubě. Proud po 10 mA jemně a po 100 mA hrubě. Nastavené a naměřené hodnoty zobrazuje jednořádkový LCD displej. Zařízení disponuje rozhraním RS232 pro komunikaci s PC, přes které je možno číst a nastavovat parametry zdroje. Přizpůsobení napěťových úrovní mezi zdrojem a sériovým rozhraním je zprostředkováno pomocí IO MAX232.

Bližší informace k regulovatelnému zdroji jsou umístěny v elektronické podobě na CD v adresáři Zdroj.

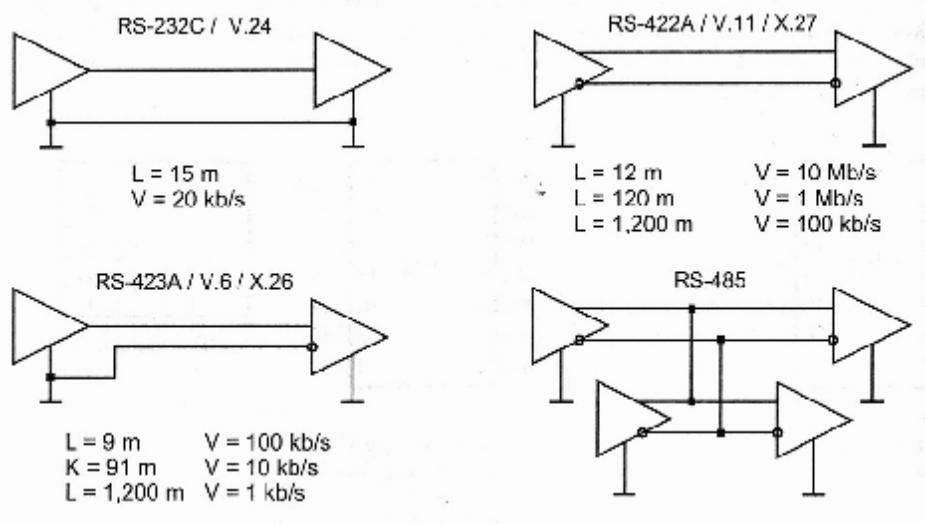


Obrázek 1: Blokové schéma zdroje

Obrázek 2: Digitálně regulovatelný laboratorní zdroj

2 Sériové rozhraní

Sériové rozhraní je jedním z nejstarších komunikačních rozhraní. V oblasti domácích PC toto rozhraní ustoupilo novému univerzálnímu sériovému rozhraní – USB, které nalezneme na převážné většině nových zařízení připojitelných k PC. V průmyslové oblasti je RS232 (včetně souvisejících rozhraní RS422, RS423 a RS485), stále aktuálním rozhraním. Tato rozhraní se používají k připojení různých periferních a komunikačních zařízení, ke komunikaci s různými technologickými přístroji, ovládacím a měřícím vybavením, k připojení programátorů integrovaných obvodů a k připojení nejrůznějších testovacích přípravků. U mnohých starších zařízení též nalezneme RS232 jako komunikační rozhraní. Obrázek 3 shrnuje přehled standardů sériového rozhraní a jeho základní parametry.



Obrázek 3: Standardy sériového rozhraní [1]

2.1 Port COM

Komunikační port COM je součástí počítačů PC již od prvních modelů. Tento port poskytuje asynchronní přenos dat s použitím standardu RS232C. Implementace portů COM je založena na integrovaných obvodech UART (Universal Asynchronous Receiver-Transmitter – univerzální asynchronní přijímač a vysílač), které jsou kompatibilní se softwarovým modelem i8250/16450/16550.

Každý port zabírá 8 sousedních 8bitových registrů ve vstupně-výstupním adresovém prostoru a může mít standardní základní adresu: 3F8h (COM1), 2F8h (COM2), 3E8h (COM3) a 2E8h (COM4). Porty mohou generovat hardwarová přerušení IRQ4 (obvykle se používá pro COM1 a COM3) IRQ3 (slouží pro COM2 a COM4). Z hardwarové stránky mají porty linky pro sériové vysílání a příjem dat. Jsou také vybaveny sadou linek pro signály řízení a sledování stavu, které vyhovují standardu RS232C. Porty COM mají na zadním panelu PC externí konektory DB25P nebo DB9P, což jsou samčí zástrčky. Všechny externí signály portu jsou bipolární (viz. kapitola 2.6). Neexistuje žádné galvanické oddělení. Uzemněné obvody připojeného zařízení jsou připojeny k uzemněným obvodům počítače. Rychlost přenosu dat může dosáhnout 115 200 b/s.

Až čtyři sériové porty COM1–COM4 lze podporovat na úrovni BIOSu počítače. Počítače třídy AT mají zpravidla 2 porty. Služba Int14h BIOSu zajišťuje inicializaci portu, vstup a výstup znaků (bez použití přerušení) a dotazování stavu. Pomocí standardních volání Int14h lze naprogramovat rychlost přenosu dat tak, aby spadala do rozsahu 110–9 600 b/s (méně než jsou skutečné možnosti portu). Kvůli zvýšení propustnosti se interakce aplikačního softwaru s portem na úrovni registrů používá velmi často. Předpokladem je, aby byl hardware portu COM kompatibilní se softwarovým modelem i8250/16450/ 16550.

Název portu označuje jeho hlavní účel: připojit zařízení (jako je například modem), které umožňuje počítači komunikovat s jinými počítači, sítěmi a periferními zařízeními. Přímo k portu lze připojit periferní zařízení se sériovým rozhraním – tiskárny, plottery atd. Port COM se často používal k připojení myši a přímému propojení dvou počítačů. Port COM také umožňuje připojit k počítači elektronické klíče. Pomocí jednoho z portů lze realizovat bezdrátovou infračervenou komunikaci s periferními zařízeními.

„Klasický” port COM umožňoval pouze softwarově řízené přenosy dat. Procesor musel k přenosu každého bajtu provést několik instrukcí. Moderní porty mají několik datových vyrovnávacích pamětí FIFO a umožňují přenos pomocí kanálu DMA. Tím výrazně klesá zatížení procesoru. Tento faktor je zvláště důležitý při vysokých přenosových rychlostech.

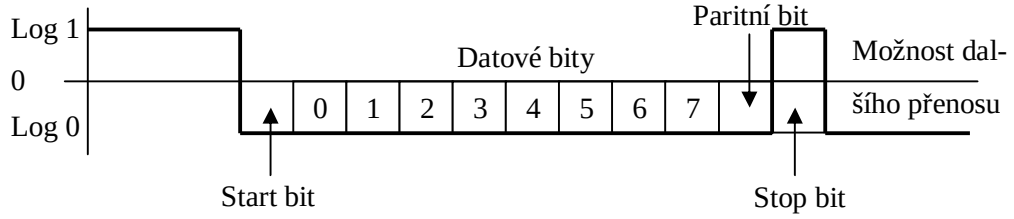
Starší základní desky mají dva vestavěné adaptéry portu COM. Novější základní desky mají většinou pouze port jeden. Pokud je požadován vetší počet rozhraní, lze instalovat další adaptéry, označované jako multiplexory.

U notebooků se v současné době port COM nepoužívá vůbec. V takovém případě je možné zakoupit převodník z USB na RS232, kde po jednoduché instalaci ovladačů můžeme plně využívat možností RS232.

2.2 Synchronní a asynchronní přenos dat

Při sériovém přenosu dat rozlišujeme synchronní a asynchronní výměnu dat. Při synchronním přenosu je přes samostatný vodič přenášen ještě jeden nebo několik signálů, které udávají, kdy je příští bit na datovém vodiči platný. Tyto signály mohou být tvořeny taktovacími signály ze zdroje taktovacího impulzu nebo handshakovými signály typu Request (žádost) a Acknowledge (potvrzení). Velkou výhodou synchronního přenosu je, že přijímač může reagovat na různé taktovací rychlosti, dokud není překročena jeho maximální frekvence, a to tak, že jednoduše zaznamenává například přechod low-high u taktovacího signálu.

Asynchronní režim sériového přenosu dat je orientován na bajty (na znaky): minimální přenesená jednotka informace je jeden bajt (jeden znak).



Obrázek 4: Formát asynchronního přenosu RS232C

Na obrázku 4 je vysvětlen způsob přenosu bajtů. Přenos každého bajtu začíná startovacím bitem, který přijímači signalizuje začátek přenosu. Poté následují datové bity a případný paritní bit. Přenos končí stopbitem, který zajišťuje prodlevu mezi přenosy. Startovací bit dalšího bajtu je odeslán kdykoliv po stopbitu předchozího bajtu. To znamená, že mezi přenosy mohou být libovolně dlouhé pauzy. Startovací bit, který má vždy přesně definovanou hodnotu (hodnotu log.

nuly), umožňuje snadno synchronizovat přijímač pomocí signálu z vysílače. Rozumí se, že přijímač i vysílač pracují se stejnou přenosovou rychlostí. Interní generátor synchronizace v přijímači používá čítač dělící referenční frekvenci, který je nastaven na nulu v okamžiku, kdy přijme začátek startovacího bitu. Tento čítač generuje interní vzorkovací impulzy umístěné doprostřed bitových intervalů, což umožňuje příjem dat i tehdy, jestliže se přenosové rychlosti přijímače i vysílače mírně liší. Při přenosu osmi datových bitů, jednoho paritního a jednoho stopbitu nesmí přípustná frekvenční tolerance, při které budou data správně detekována, překročit 5 %. Při započtení fázového zkreslení je ve skutečnosti povolena ještě nižší frekvenční tolerance. Při zvyšování přenosové rychlosti rostou požadavky na sladění frekvencí přijímače a vysílače. Formát asynchronního přenosu umožňuje detekovat potenciální chyby přenosu.

Pokud byl přijat logický přechod signalizující začátek přenosu, ale vzorkovací impulz by zachytil logickou jedničku, považuje přijímač startovací bit za neplatný a vrátí se do stavu čekání. Přijímač tuto chybu nemusí oznamovat.

Pokud se během časového intervalu, vyhrazeného pro stopbit, objeví logická nula, zaznamená se chyba stop bitu. Pokud se uplatňuje kontrola parity, je po odeslání datových bitů odeslán bit kontroly parity. Tento bit doplňuje počet logických jedniček datových bitů na sudou nebo lichou hodnotu v závislosti na předchozím nastavení. V případě přijetí chybné hodnoty paritního bitu je zaznamenána chyba.

Kontrola formátu umožňuje detekci poškození linky. Platí pravidlo, že přijímač při výskytu poškození linky „vidí“ logickou nulu. Tato logická nula je nejdříve považována za startovací bit a nulové datové bity, ale potom je aktivována kontrola bitu.

Pro asynchronní režim je přijata řada standardních přenosových rychlostí: 50, 75, 110, 150, 300, 600, 1 200, 2 400, 4 800, 9 600, 19 200, 38 400, 57 600 a 115 200 bitů za sekundu.

2.3 Parita

Parita je jednoduchý, ale málo bezpečný způsob ochrany proti chybám přenosu. Pomocí parity je možno spolehlivě rozeznat jen jednoduché bitové chyby.

Chyby bloků (burst errors) s několika poškozenými bity s 50% pravděpodobností zaznamenány nebudou. Parita je tedy vhodná jen pro krátké přenosové cesty, které nejsou příliš náchylné k výskytu chyb. Pro jiná použití jsou mnohem spolehlivější CRC kódy, které jsou mnohem složitější na výpočet. Výhodou parity však je, že téměř všechny prvky sériových rozhraní podporují vytváření a kontrolu paritních bitů hardwarově.

Celkem rozlišujeme pět různých typů parity:

Žádná parita: Není připojen žádný paritní bit.
Sudá parita: Paritní bit je vložen tak, aby byl v datových bitech a paritním bitu celkem sudý počet jedniček.
Lichá parita: Paritní bit je vložen tak, aby byl v datových bitech a paritním bitu celkem lichý počet jedniček.
Mark: Paritní bit je nastaven vždy na hodnotu „1".
Space: Paritní bit je nastaven vždy na hodnotu „0".

Mark a space se hodí spíše jen k zaznamenání chyb v paritním bitu samotném. Když se vyskytne SDU s vymazaným nebo vloženým paritním bitem a měla by být přítomna parita Mark (resp. Space), je v paritním bitu chyba. O správnosti datových bitů to samozřejmě nic nevypovídá. Datové bity mohou být naopak zničené, ale přijímač si toho nevšimne, je-li paritní bit nepoškozený. Mark a Space tedy mají jen velmi malou až zbytečnou hodnotu.

2.4 Přenosová rychlost (baud rate)

Další veličina v souvislosti se sériovým přenosem dat, kde často dochází k chybám, je přenosová rychlost (baud rate). Pojmenována byla podle francouzského matematika J. M. E. Baudota a udává pouze počet změn signálů přenosového kanálu za sekundu. Protože u obvyklých sériových rozhraní jsou změny signálů časově stejné a probíhá velmi jednoduché binární kódování dat – logicky vysoká úroveň odpovídající paritě Mark odpovídá „1", logicky nízká úroveň odpovídající Space odpovídá „0", je zde přenosová rychlost rovna počtu přenášených bitů za sekundu (bps), pokud zahrneme také start bit, paritní bit a stop bity. Při výkonném kódování s metodou komprese dat, jaké se použí-

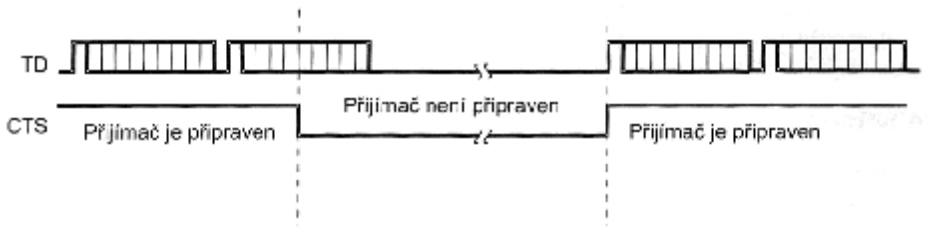
vá například u modemů, je dosahováno datových rychlostí (v bps), které výrazně převyšují baud rate.

2.5 Řízení toku dat

K řízení toku dat lze použít dva typy protokolu: hardwarový a softwarový. Řízení toku dat se někdy zaměňuje s handshakingkem neboli potvrzováním. Handshaking znamená vyjednávání každého základního kroku protokolu s partnerem. Potvrzení je odeslání oznámení o příjmu kvanta dat (bajtu, rámce, paketu). Řízení toku předpokládá, že se budou odesílat oznámení o tom, zda je či není možný následný příjem dat. Poměrně často je řízení toku založeno na mechanismu handshaking nebo na potvrzování.

2.5.1 Hardwarové řízení toku dat

Protokol hardwarového řízení toku používá signál CTS (Clear To Send), který umožňuje zastavit odesílání dat, pokud je není přijímač připraven přijmout.



Obrázek 5: Hardwarové řízení toku [1]

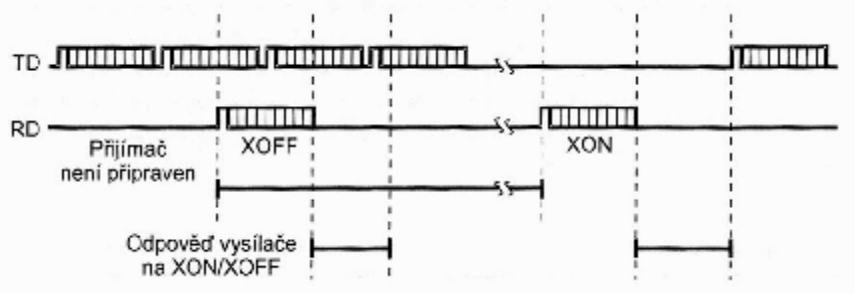
Vysílač „uvolní“ další bajt (TD) pouze tehdy, je-li nastavena linka CTS. Pomocí signálu CTS není možné zastavit bajt, jehož přenos již začal – tím je zajištěno že nebude poškozen již přenášený bajt. Hardwarový protokol zajišťuje nejrychlejší reakci vysílače na stav přijímače. Integrované obvody asynchronních vysílačů mají minimálně dva přijímací registry: posuvný registr pro příjem dalšího paketu a úložný registr, ze kterého je přijatý bajt načítán. Díky tomu lze implementovat výměnu založenou na hardwarovém protokolu bez ztráty dat.

Hardwarový protokol je praktický při připojení tiskáren a plotterů, pokud jej podporují. V případě přímého propojení dvou počítačů (bez modemů) hardwarový protokol vyžaduje, aby byly linky RTS a CTS překříženy.

U přímého připojení musí mít vysílací terminál svou linku CTS přepnutou do stavu ON (vzájemným propojením svých linek RTS a CTS). V opačném případě nebude vysílač odesílat data.

2.5.2 Softwarové řízení toku dat

Softwarový protokol řízení toku dat XON/XOFF vychází z předpokladu, že je k dispozici obousměrný kanál pro přenos dat. Protokol funguje následujícím způsobem: pokud přijímací zařízení zjistí, že z určitého důvodu nemůže nadále přijímat data, odešle pomocí zpětného sériového kanálu symbol bajtu XOFF (13h). Po příjmu tohoto signálu protější zařízení pozastaví přenos. Když přijímací zařízení přejde do stavu, ve kterém může opět přijímat data, odešle znak XON (11h). Protější zařízení po jeho příjmu obnoví přenos. V porovnání s hardwarově implementovaným protokolem se čas odpovědi vysílače na změnu stavu přijímače zvyšuje minimálně o dobu přenosu signálu (XON nebo XOFF) a navíc o interval, než vysílač na příjem znaku zareaguje (obrázek 6). Z toho vyplývá, že bez ztráty dat může fungovat pouze přijímač s dodatečnou vyrovnávací pamětí na přijatá data. Takový přijímač navíc musí předem informovat o tom, že nemůže pokračovat v příjmu (dokud je ve vyrovnávací paměti místo).



Obrázek 6: Softwarové řízení toku dat [1]

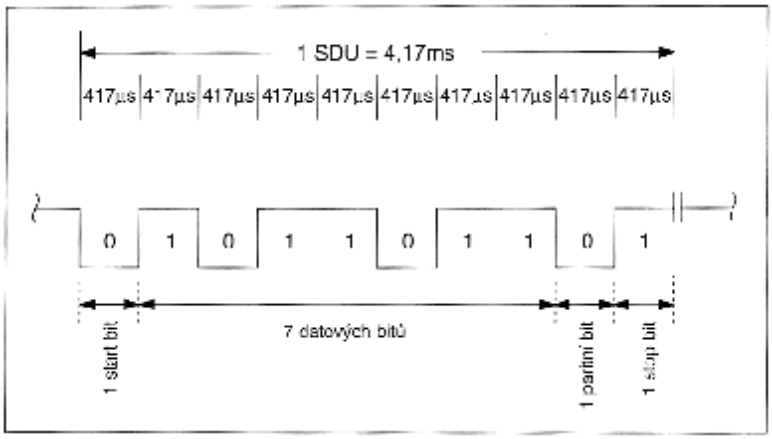
Výhoda softwarového řízení toku spočívá v tom, že není nutné přenášet řídící signály rozhraní. Nejjednodušší kabel pro obousměrnou komunikaci může ob-

sahovat pouze tři vodiče. K nedostatkům patří kromě potřeby vyrovnávací paměti a dlouhé reakční doby (která snižuje celkovou účinnost kanálu kvůli čekání na signál XON) také obtížná implementace plně duplexního režimu výměny dat.

Kromě těchto dvou běžně používaných standardních protokolů, které podporují operačním systémem a většinou periferních zařízení, existují vlastní protokoly vyžadující speciální podporu.

2.6 Serializace dat a SDU

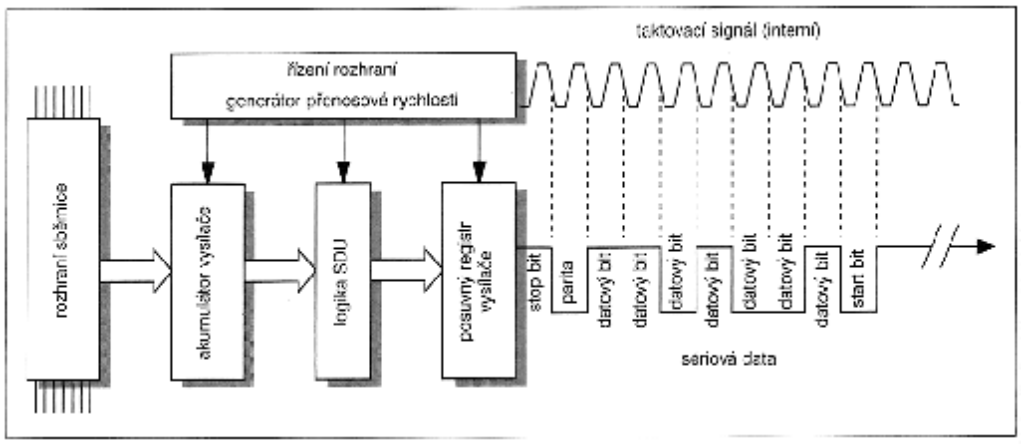
SDU – Serial Data Unit (sériová datová jednotka) – je nejmenší množina dat, která je předávána sériovým rozhraním. Před uskutečněním asynchronního sériového přenosu dat mezi dvěma jednotkami, musejí být vysílač a přijímač nastaveny na stejné formáty. Je nutné uvést počet datových bitů, jestli bude použita parita, případně její druh a počet stop bitů. Platné hodnoty pro počet datových bitů jsou pět, šest, sedm a osm. Jestliže bylo zvoleno pět datových bitů, je čipem sériového rozhraní automaticky přidán jeden a půl stop bitu. Tím je myšleno, že časové trvání stop bitů činí jeden a půl normálního bitu. Pro ukončení přenosu je možno volit jeden, jeden a půl, nebo dva stop bity. Baud rate pro vysílač a přijímač musí být nastavena na stejnou hodnotu.



Obrázek 7: SDU [3]

Na obrázku 7 je zobrazena SDU s jedním start bitem, sedmi datovými bity, lichou paritou a jedním stop bitem. Start bit má vždy hodnotu 10 (to odpovídá space) a stop bit či bity mají vždy hodnotu 1 (to odpovídá mark).

Obrázek 8 znázorňuje schéma pro vytváření sériového sledu dat. Čip rozhraní má pro vysílání dat akumulátor vysílače, který nejprve přijme datový bajt od CPU. Logika SDU připojí podle zvoleného datového formátu start bit před nastavený počet datových bitů, případně vypočítá paritní bit a spolu s nastavenými stop bity sestaví datovou jednotku. Takto vytvořený sled dat je přenesen do posuvného registru vysílače. Posuvný registr vysílače je nyní generátorem taktu řízen v taktu baud rate a jednotlivé bity jsou předány sériovému výstupu počínaje start bitem. Následují datové bity v pořadí o nejnižšího po nejvyšší. Přenos je zakončen stopbitem.



Obrázek 8: Převod a vysílání sériových dat [3]

Příjem proběhne zcela opačně. Start bit s hodnotou logické 1, což odpovídá space, je jednoznačný signál pro přicházející sled dat, protože přijímací vodič je vysílačem v normálním případě udržován na mark. Start bit tedy působí jako spínací impuls a spustí přijímač v čipu sériového rozhraní. Sériové bity jsou v taktu nastavené baud rate zavedeny do posuvného registru přijímače. To znamená, že do posuvného registru přijímače jsou nejprve zavedeny nižší a pak vyšší datové bity. Logika přijímače oddělí od přijatých bitů start bit, paritní bit (pokud je přítomen) a stop bit či bity, vypočítá případnou paritu přijatých bitů a porovná ji s nastavenou paritou. Následně jsou extrahované datové bity přene-

seny do vyrovnávacího registru přijímače, kde je CPU může načíst jako datový bajt. Schéma příjmu je totožné s obrázkem 8 s tím rozdílem, že data prochází opačným směrem. Tzn. že sériová data jsou přijata posuvným registrem v tomto případě přijímače, zpracována logikou SDU, předána akumulátoru přijímače a následně předána rozhraní sběrnice.

Z popisu průběhu vysílání a příjmu je zřejmé, že u vysílače a přijímače musí být nastavena stejná baud rate. Kromě toho musejí být stejné i nastavené datové formáty – počet datových bitů, parita, a počet stop bitů, protože jinak přijímač pravděpodobně složí jiný bajt, než jaký byl vysílači předán k vysílání.

2.7 Rozhraní RS323C

Obvyklá rozhraní PC pro sériovou výměnu dat vyhovují standardu RS232C od EIA (Electronic Industrie Association, vydavatel specifikace RS232C). V Evropě se často setkáme také s označením V.24, které je zcela rovnocenné s RS232C.

Standard definuje mechanické, elektrické a logické rozhraní mezi koncovým zařízením přenosu dat DTE (Data Terminal Equipment) a zařízením pro přenos dat DCE (Data Carrier Equipment). Koncové zařízení přenosu dat DTE je obvykle tvořeno počítačem a zařízení pro přenos dat DCE je například modem nebo.

Standard RS232C předpokládá 25 vodičů mezi DTE a DCE, a tedy konektor s 25 vývody. Většina vývodů je však rezervována pro synchronní přenos dat. Sériová asynchronní výměna dat je však u PC považována za standard. Sériové rozhraní v PC tedy představuje jen podmnožinu možností stanovených ve specifikaci RS232C, resp. V.24. Následující část textu bude popisovat pouze sériové rozhraní PC.

V PC je zapotřebí jen jedenáct signálů specifikace RS232C. Navíc IBM definovalo připojení s 9 vývody, u něhož chybí dva vodiče RS232C, které zde v normálním případě jsou. Tabulka 1 popisuje obsazení a signály pro konektory s 25 a 9 vývody, přičemž konektor s 9 vývody je dnes u PC obvyklý.

Tabulka 1: Zapojení konektorů RS232C s 9 a 25 vývody

DB25P konektor	DB9P konektor	Funkce	Směr
1	–	ochranná zem	–
2	3	vysílaná data – TD	DTE→DCE
3	2	přijatá data – RD	DCE→DTE
4	7	RTS	DTE→DCE
5	8	CTS	DCE→DTE
6	6	DSR	DCE→DTE
7	5	signálová zem	–
8	1	DCD	DCE→DTE
20	4	DTR	DTE→DCE
22	9	RI	DCE→DTE
23	–	DRSD	DCE→DTE

Zdroj: Specifikace RS232C

U připojení s 9 vývody chybí ochranná zem a signál pro rychlost datového signálu. Zbývajících devět signálů ale postačuje pro sériovou, asynchronní výměnu dat mezi DTE a DCE podle standardu RS232C. Oba vývody 3/2 a 2/3 přitom přenášejí datové signály, zbývající kontakty pak řídicí signály.

Pro řízení přenosu dat mezi DTE a DCE je rozhodujících pět řídicích signálů RTS (Request to Send, vývod 4/7), CTS (Clear to Send, vývod 5/8), DCD (Data Carrier Detect, vývod 8/1), DSR (Data Set Ready, vývod 6/6) a DTR (Data Terminal Ready, vývod 20/4).

RTS (Request To Send)

Tento signál od DTE instruuje DCE, aby se nastavilo na přenos dat z DTE na DCE. DTE tedy tímto způsobem signalizuje DCE, že chce provést výstup dat, která má DCE přijmout.

CTS (Clear To Send)

Tento signál (od DCE) indikuje DTE, že DCE je připraveno přijmout data od DTE. DCE aktivuje signál CTS obvykle jako reakci na aktivaci signálu RTS od DTE. Jestliže DCE aktivovalo signál CTS, může DTE začít s výstupem dat.

RTS a CTS slouží u poloduplexního spojení k rozdělení rolí obou komunikačních partnerů jako vysílače a přijímače a přepnutí obou partnerů na vysílání a příjem. RTS a CTS tak tvoří handshakové signály.

DCD (Data Carrier Detect)

DCE aktivuje signál DCD, když zaznamená nosný signál od cíle a je vytvořeno spojení. DCD zůstane aktivní po celou dobu spojení. U poloduplexních modemů vysílá signál DCD jen přijímající DCE.

DSR (Data Set Ready)

DCE (obvykle modem) pomocí aktivního signálu DSR oznamuje DTE, že je zapnuté, že ukončilo přípravy pro spojení s cílem a může s DTE komunikovat. Data Set Ready přitom obecně označuje externí koncové zařízení přenosu dat.

DTR (Data Terminal Ready)

Signál od DTE indikuje obecnou připravenost DTE a je obvykle aktivován při zapnutí DTE. Když je spojení mezi DTE a DCE jednou vytvořeno, musí DTR zůstat aktivní během celé doby spojení. DTR a DSR jsou tedy odpovědné za vytvoření spojení, RTS a CTS zase za přenos dat – a za směr přenosu u poloduplexního spojení. Bez aktivního signálu DTR nemají RTS a CTS žádné účinky, DCE nijak nereaguje na řídicí signály a nevysílá ani nepřijímá data. DTR tak prakticky představuje hlavní spínač. Deaktivace DTR nebo také DSR spojení přeruší.

RI (Ring Indicator)

Signál RI oznamuje DTE, že k DCE dorazil zvukový signál. Tak je tomu např. tehdy, když externí počítačový systém zvolí modem prostřednictvím telefonní linky. Modem může signál RI aktivovat, i když je DTR neaktivní, tj. RI pracuje nezávisle na DTR.

DSRD (Data Signal Rate Detektor)

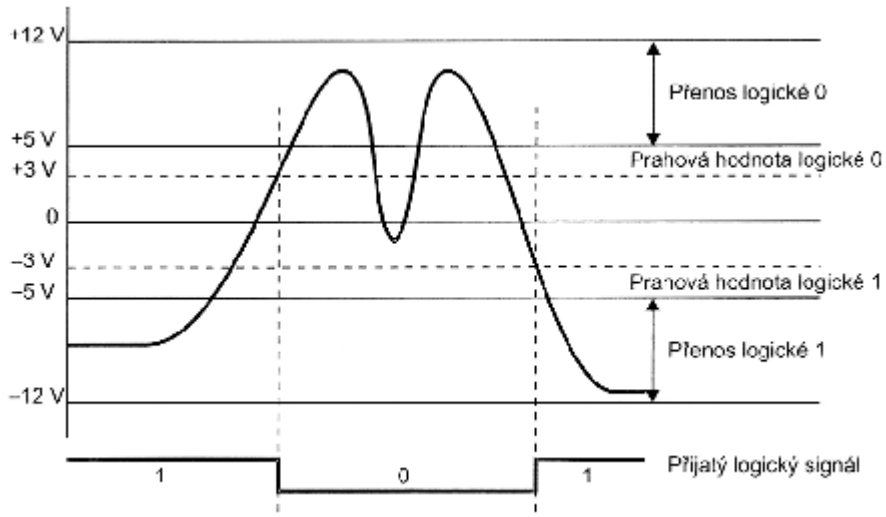
Připojení s 25 vývody může následně přenášet ještě signál DSRD (Data Signal Rate Detector, vývod 23), který umožňuje přepnutí mezi dvěma různými pře-

nosovými rychlostmi. Signál může být mezi DTE a DCE přenášen oběma směry. Tak je např. možné, aby zařízení DTE přikázalo DCE, aby zvolilo vysokou přenosovou rychlost nebo aby DCE sdělilo DTE přenosovou rychlost přenosového kanálu. Použití pěti uvedených signálů se liší podle typu spojení.

2.7.1 Logické úrovně RS232C

Na rozdíl od logických signálů, které se v normálním případě v oblasti PC používají, jsou signály RS232C bipolární. To znamená, že nízkou logickou úroveň neudává úroveň 0 V, která odpovídá zemi GND, nýbrž záporná hodnota napětí. Logická 1 je tedy oproti zemi znázorněna jako kladné napětí, logická 0 jako záporné napětí. Pro výstupní signály se hodnota napětí pro logickou 1 pohybuje mezi +5 V a +15 V, pro logickou 0 mezi −5 V a −15 V. U vstupních signálů je log. 1 znázorněna napětím mezi +3 V a +15 V a log. 0 pomocí napětí −3 V a −15 V. Tyto hodnoty se označují jako úrovně signálů EIA.

Široké spektrum napětí pro logické hodnoty umožňuje bezproblémový přenos dat také na větší vzdálenosti. V PC jsou logické úrovně reprezentovány hodnotami do ±12 V, protože napájecí zdroje v PC poskytují napětí do ±12 V.



Obrázek 9: Napěťové úrovně signálů RS232C [3]

2.8 UART – univerzální asynchronní přijímač-vysílač

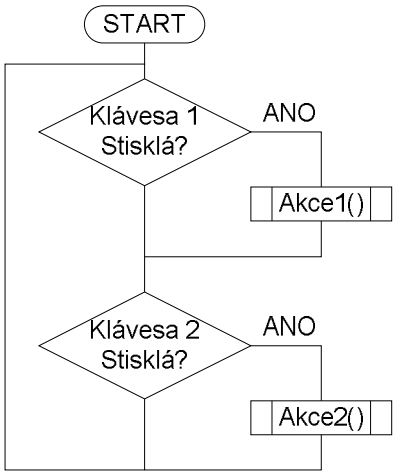
Převod paralelních dat na sériová za účelem přenosu a opačný převod při příjmu zajišťují v portech COM integrované obvody UART. Stejné integrované obvody generují a zpracovávají řídící signály rozhraní. Porty COM počítačů IBM PC XT/AT jsou založeny na integrovaných obvodech, které jsou na úrovni registrů kompatibilní s řadou UART i8250: 8250/16450/16550A. Tato řada je vylepšením původního modelu, což vedlo ke zvýšení provozní rychlosti. Při intenzivní výměně dat se snížila spotřeba energie a zátěž procesoru.

3 Využiti Borland Delphi pro vývoj řídících aplikací

Pro programování řídících aplikací je vhodné využít tzv. událostí, což je prvek, kterým disponuje jazyk Object Pascal, které nám umožní efektivně reagovat na přijatá data při minimálním vytížení procesoru. V aplikacích pak není nutné se cyklicky dotazovat, zda přišla data na porty PC. Stačí pouze reagovat na událost která nám oznámí, že byla přijata data. V obsloužení události pak přijatá data zpracujeme.

3.1 Události v Delphi (events)

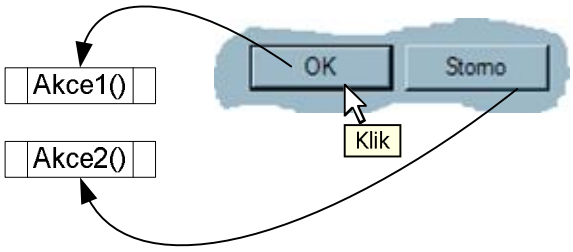
Události jsou procedury, které nám umožní obsloužit určitou situaci, která nastane za běhu programu. V případě konzolových aplikací se využívalo především sekvenčního přístupu, kde se po spuštění procházela určitá sekvence příkazů, která se na základě uživatelovy volby dále větvila. Nevýhodou tohoto způsobu bylo velké vytížení CPU.



Obrázek 10: Sekvenční přístup

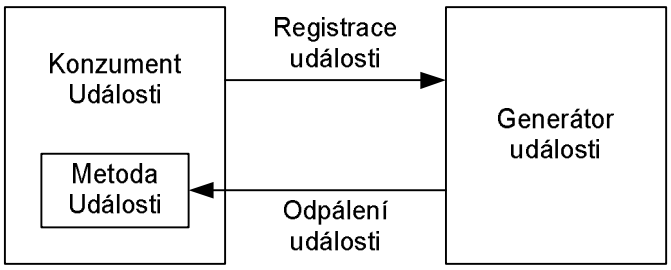
V aplikacích s uživatelským rozhraním – GUI (např. Delphi) se píše obslužný kód do procedur a funkcí, které ošetřují jednotlivé události (jsou označovány jako Event Handlers). Například při stisku tlačítka (Click) je generována udá-

lost OnClick. Běh programu si tedy lze představit jako úvodní inicializaci a následné čekání na události, které realizuje OS svými prostředky. Tato činnost počítač (operační systém) nezatěžuje.



Obrázek 11: Vyvolání události

Pokud uživatel stiskne tlačítko, nastane událost OnClick na základě které je vykonán kód napsaný v proceduře (např. `TlačítkoOnClick()`) příslušného tlačítka. Vývojové prostředí nám nabízí velké množství událostí, které většinou postačí pro vývoj programu. V případě specifických aplikací může nastat situace, kdy potřebujeme vytvořit vlastní událost. Události se netýkají jen grafického rozhraní, ale mohou sloužit i jako komunikační nástroj mezi jednotlivými objekty. Když dojde k situaci, kterou chceme sledovat v jednom objektu, je o této události informován druhý objekt, který má možnost na tuto situaci reagovat. Model této situace je vysvětlen na obrázku 12. Konzument a generátor události jsou objekty. Pro příjem události je nutné se zaregistrovat k odběru události. Když nastane událost (odpálení události) je zavolána metoda (event handler), která událost obslouží.

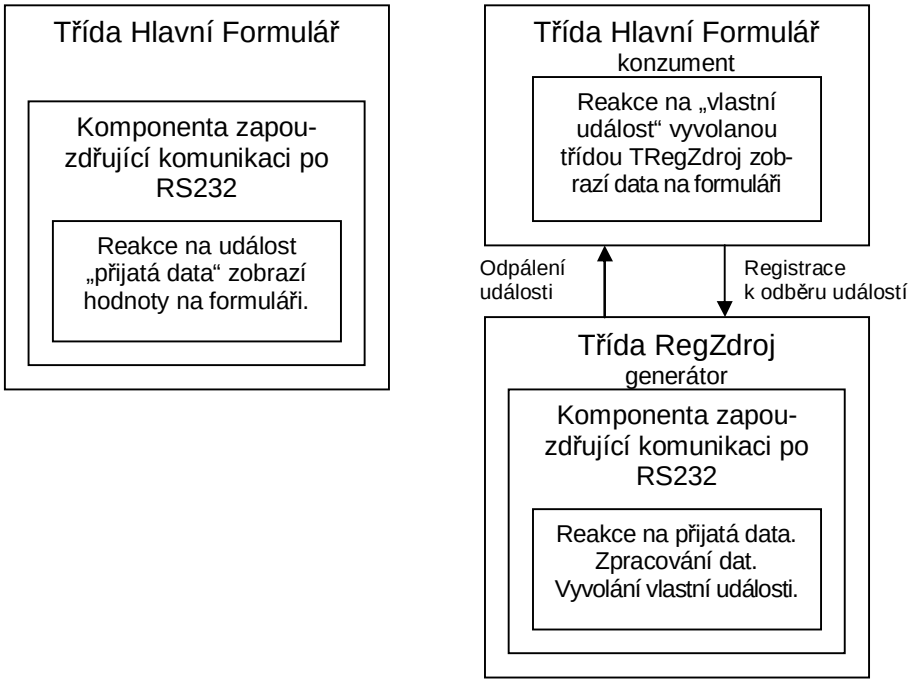


Obrázek 12: Příjem a obsloužení událostí

3.2 Využití událostí v řídící aplikaci

Jak již bylo zmíněno práce s událostmi se netýká jen programování v grafickém režimu, ale je možné ji využít i pro komunikaci mezi objekty. Libovolná třída může generovat vlastní událost. Např. jedna třída oznámí druhé: „Právě jsem zpracovala data, tak je zobraz.“

Tento princip se uplatnil v rámci psaní řídící aplikace k regulovatelnému zdroji. V rámci zásad OOP bylo nutné oddělit vizuální návrh (formulář) od ovládání regulovatelného zdroje. Tímto vznikl problém. Oddělením komponenty, zapouzdřující komunikaci po sériovém portu, od formuláře nebylo možné reagovat na příchozí data v hlavním formuláři (např. zobrazit aktuální výstupní hodnoty).



Obrázek 13: Aplikace událostí v rámci bakalářské práce.

Bylo tedy nutné vytvořit vlastní událost, která by oznámila formuláři, že má zobrazit nové výstupní hodnoty. Levá strana obrázku 13 znázorňuje předávání dat v rámci jedné třídy. Po oddělení vizuálního návrhu od obsluhy zdroje bylo třeba zařídit „dorozumívání“ mezi objekty – pravá strana obrázku 13.

Pro aktualizaci zobrazení dat v hlavním formuláři byla vytvořena vlastní událost, která je vyvolána v případě, že třída TRegZdroj zpracuje přijatá data ze sériového portu (zpracuje data od třídy TVaComm).

3.3 Vytvoření vlastní události

Následující text popisuje způsob vytvoření vlastní události. V adresáři Události na CD se nachází ukázková aplikace.

1) **Vytvoření události (pomocí property) v rámci třídy, která bude generovat vlastní událost**
   - Property OnMyEvent je generátor události.
   - Při spuštení aplikace je atributu FOnMyEvent přiřazena adresa metody, která obslouží událost (Event handler)
   - Metoda BtnVyvolejUdalostClick (což je v tomto případě kliknutí na tlačítko) odpálí událost. Tím je vyvolána metoda obsluhy.

```
type TUdalost = class(TForm)
    BtnVyvolejUdalost: TButton;
    procedure BtnVyvolejUdalostClick(Sender: TObject);
  private
     FOnMyEvent:TNotifyEvent;*
     procedure SetOnMyEvent(Value:TNotifyEvent);*
  public
     property OnMyEvent:TNotifyEvent  read FOnMyEvent
                                      write SetOnMyEvent;*
end;

procedure TUdalost.SetOnMyEvent(Value:TNotifyEvent);*
begin
  FOnMyEvent:=Value;
end;

// odpalení události
procedure TUdalost.BtnVyvolejUdalostClick(Sender: TObject);
begin
  // test přihlášení k odběru události a následné odpálení
  // události; pokud neni konzument přihlášen je gen. výjimka
  if Assigned(FOnMyEvent) then FOnMyevent(self);
end;

// * deklarace vlastní události
```

2) Přihlášení se k odběru událostí – objekt konzument události

```
type TKonzumentUdalosti = class(TForm)
    LblUdalost: TLabel;
    procedure FormCreate(Sender: TObject);
  private
    Udalost:TUdalost;
    procedure ObsluhaUdalosti(Sender:TObject);
  public
end;

procedure KonzumentUdalosti.FormCreate(Sender: TObject);
begin
  //vytvoreni instance tridy TUdalost
  Udalost:=TUdalost.Create(Self);
  //prihlaseni k odberu udalosti
  Udalost.OnMyEvent:=Self.MetodaUdalosti;
  Udalost.Show;
end;

procedure TKonzumentUdalosti.MetodaUdalosti(Sender:TObject);
begin
  LblUdalost.Caption:='Nastala udalost!';
  ShowMessage('Form1.Nastala udalost!');
end;
```

3) Vyvolání události

```
// event handler
procedure TKozumentUdalosti.Obsluha(Sender:TObject);
begin
  LblUdalost.Caption:='Nastala udalost';
  ShowMessage('Nastala udalost');
end;
```

4 Přístup k portům v Borland Delphi

Tato kapitola popisuje problematiku přístupu k portům PC v prostředí Borland Delphi.

4.1 Přímý přístup k portům

Chráněný režim procesorů Intel poskytuje 4 různé úrovně přístupu, které se označují ring 0, ring 1, ring 2 a ring 3. Ring 0 je nejvyšší úroveň oprávnění a ring 3 nejnižší. Operační systémy Windows 95 a Windows NT používají pouze ring 0 a ring 3.

Je-li kód prováděn v ring 0, může obsahovat libovolné instrukce, tedy i instrukce přístupu ke vstupně výstupním portům. Oproti tomu ring 3 je v tomto přístupu velmi omezen.

Pod Windows běží v ring 0 pouze jádro operačního systému a ovladače zařízení. Všechny uživatelské programy běží v ring 3. Proto je jejich schopnost ve smyslu vstupně/výstupních operací značně omezena.

Windows 95, 98 povolují přístup na vstupně/výstupní porty i pro ring 3, Windows vycházející z koncepce NT tyto operace zase zakazují. Proto byla z Delphi (překladači generující kód pro operační systém Windows) vyjmuta proměnná Port, která umožňovala přístup na porty aplikacím přeloženým v prostředí Turbo Pascal, později Borland Pascal pro operační systém MS DOS.

Novější verze operačního systému Windows, založené na koncepci NT, hlídají aby software nedělal co by neměl na hardwarové vrstvě a proto je nutné prostupovat k hardware korektně přes Windows API funkce.

4.1 Komponenty zapouzdřující komunikaci po RS232

Jak již bylo zmíněno, prostředí Delphi samo o sobě nepodporuje komunikaci po RS232. Je možné požít Windows API, což je velmi obsáhlá knihovna funk-

cí, která umožňuje vyvolávat jednotlivé služby operačního systému, nebo použít některou z externích komponent zapouzdřujících komunikaci po RS232, která v rámci svých metod vyvolá potřebné funkce WinAPI. Pro účely komunikace s laboratorním zdrojem postačí vhodná komponenta. K dispozici je mnoho komponent, ale ne každá je vhodná.

Při návrhu bylo vybíráno z následujících komponent zapouzdřujících komunikaci po RS232:

- Varian Async 32,
- TMS Async – vychází z Varian Async 32 (je téměř shodná)
- Comport v1.7
- CiaComPort
- ZLPortIO.

Komponenty jsou dostupné v příloze na CD v adresáři komponenty.

Bylo nutné zvolit komponentu, která umožňuje zápis na sériový port a čtení znaků ze sériového portu (některé komponenty umožňovaly pouze práci se signály sériového rozhraní). Dále bylo vhodné zvolit komponentu, která má implementovány události (reakce na příchozí data). Vybraná komponenta musela bezproblémově komunikovat se zdrojem. Těmto kritériím nejlépe vyhovovala komponenta VaComm z freeware balíčku Varian Async, se kterou zdroj bezproblémově spolupracoval.

4.2 TVaComm

Třída TVaComm zapouzdřuje komunikaci po sériovém portu – umožní aplikaci přijímat data a odesílat data. Používá klasických metod Delphi a to událostmi řízenou komunikaci, která umožňuje reagovat na události, které se stanou za běhu aplikace. Například při obdržení paketu na rozhraní nebo požadavku RTS (Request To Send – požadavek na zaslání dat). Třída podporuje detekci a práci s chybami, které vzniknou při přenosu. Jedna instance třídy umožňuje práci

s jedním portem. Pokud bychom chtěli pracovat s více porty najednou potřebovali bychom použít tolik instancí třídy, kolik portů bychom chtěli obsluhovat.

4.2.1 Instalace

Pro instalaci postačí zkopírovat soubory komponenty do zvoleného adresáře. Přidání komponenty do vývojového prostředí se provede otevřením souboru s příponou dpk, ve kterém se nachází veškeré potřebné informace o balíčku. Otevře se nové okno s informacemi a možnostmi nastavení balíčku. Komponenta se nainstaluje kliknutím na Install. Následuje přidání cesty k souborům balíčku kliknutím na Tools → Environment Options → záložka Library a zde doplníme cestu do Library path.

4.2.2 Možnosti čtení a zápisu dat

Přijatá data jsou načítána do vyrovnávací paměti. Následné zpracování dat je možné třemi metodami: `ReadBuff()`, `ReadText()`, `ReadChar()`. `ReadBuff()` načte požadované množství dat z bufferu, `ReadText()` načte veškerá data v bufferu a `ReadChar()` načte právě jeden znak.

Při použití komponenty se nejlépe osvědčila metoda `ReadText()`, která zpracuje veškerá přijatá data a vrátí je jako string. V kombinaci s reakcí na událost `OnRxChar()`, která je vyvolána při doručení každého paketu, se v ideálním případě zpracují data vždy po doručení nového paketu a vstupní buffer se vyprázdní. V případě, že se na událost `OnRxChar()` musí čekat (např. kvůli vytížení CPU) jsou data shromažďována do vyrovnávací paměti a zpracována následně.

Data je možné odesílat pomocí trojice metod `WriteBuff()`, `WriteText()`, `WriteChar()`, kde `WriteBuff()`zapíše nastavené množství dat do výstupního bufferu, `WriteText()`zapíše řetězec do výstupního bufferu a `WriteChar()` zapíše právě jeden znak do výstupního bufferu.

Pro komunikaci s regulovatelným laboratorním zdrojem spolehlivě fungoval přenos znak po znaku s pauzou 1 ms mezi jednotlivými znaky.

V případě přenosu většího množství dat, je nutné pro bezproblémový zápis a čtení dat, používat metod `ReadBuff()` a `WriteBuff()`. Společně s nimi je vhodné používat dalších metod `ReadBuffFree()`, `ReadBuffUsed()`, `WriteBuffFree()`, `WriteBuffUsed()`, pomocí kterých zjistíme nakolik jsou vyrovnávací paměti zaplněné. Při přijímání dat můžeme využít události `OnRx80Full()`, pomocí které zjistíme, že příchozí vyrovnávací paměť je již z 80 % zaplněna.

4.2.3 Použité metody

Jako každá standardní komponenta v Delphi má i tato komponenta své vlastnosti, metody a události. Vlastnosti se dále dělí na vlastnosti návrhu (design property) a vlastnosti nastavované za běhu aplikace (runtime property). Metody (funkce) vykonávají požadované operace a pomocí událostí můžeme reagovat na situace, které nastanou za běhu aplikace.

Následuje výpis vlastností, metod a událostí, se kterými bylo pracováno při návrhu.

Vlastnosti návrhu – design property

`TVaComm.BaudRate`

Nastaví přenosovou rychlost portu – Baud rate.

`TVaComm.Parity`

Nastaví paritu – žádná, sudá, lichá, mark, space.

`TVaComm.Databits`

Nastaví počet přenášených datových bitů – 4, 5, 6, 7, 8.

`TVaComm.Stopbits`

Nastaví počet stopbitů přenosu – 1, 1.5, 2.

`TVaComm.PortNum`

Nastaví číslo portu.

`TVaComm.FlowControl`

Nastavení režimu toku dat.

`TVaComm.DeviceName`

Slouží pro pojmenování příslušného jména portu. Při nastavení `PortNum=2` a property `DeviceName=COM%d` získáme `DeviceName=Com2`. Nastavením `PortNum=2` a `DeviceName=COM1` získáme `DeviceName=COM1`;

Metody

`TVaComm.Open()`

Otevře port.

`TVaComm.Close()`

Uzavře port. Metoda se volá automaticky při uzavření aplikace.

`TVaComm.Active()`

Vrací aktuální stav portu. Vrátí false pokud port není otevřen.

`TVaComm.WriteText()`

Zapíše na port řetězec znaků. V případě neúspěchu zápisu dat (pomalejší zařízení, problémy se zápisem) je vhodné zapisovat v cyklu znak po znaku s prodlevou alespoň 1 ms, aby nedocházelo ke ztrátě informací. Pokud se zápis nedaří je nutné prodloužit prodlevu mezi jednotlivými znaky.

`TVaComm.WriteChar()`

Zapíše na port jeden znak.

`TVaComm.ReadBuf()`

Načte požadované množství dat z vyrovnávací paměti.

`TVaComm.ReadText()`

Vrátí všechna přijatá data z vyrovnávací paměti jako řetězec.

`TVaComm.ReadChar()`

Vrátí právě jeden znak z vyrovnávací paměti. Pokud nejsou ve vyrovnávací paměti žádné znaky, vrátí false.

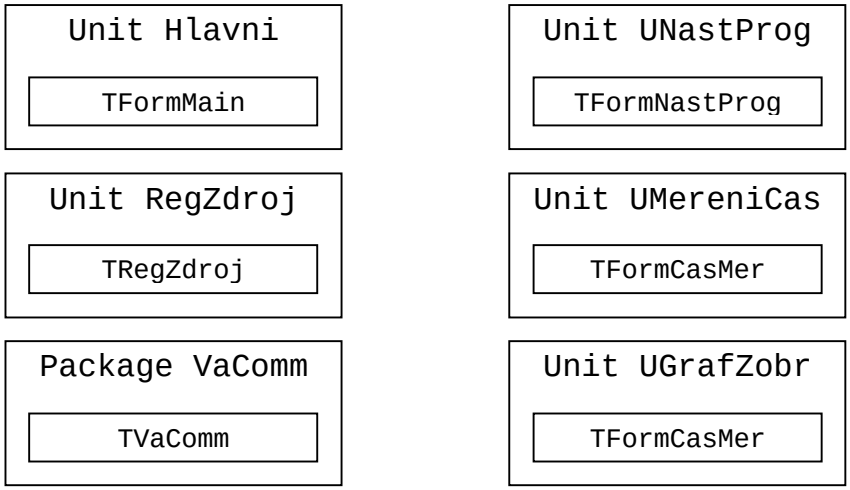
Události

`TVaComm.OnRxChar()`

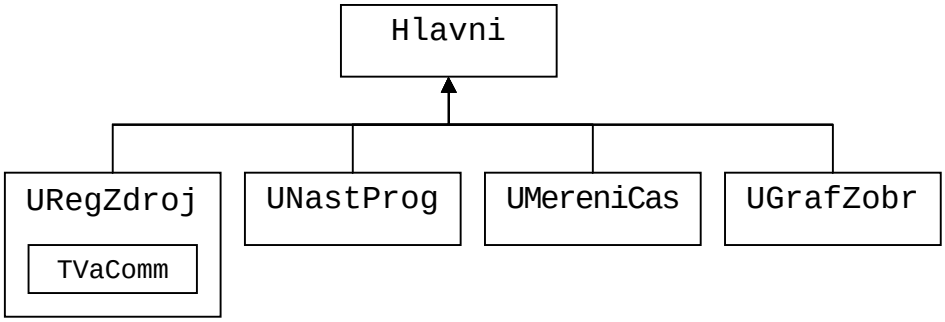
Událost je vyvolána při doručení paketu.

5 Ovládací program

Program se skládá z hlavní unity, ve které je uložen hlavní formulář, 4 vedlejších unit a komponenty VaComm zapouzdřující komunikaci po sériovém portu (obrázek 14).



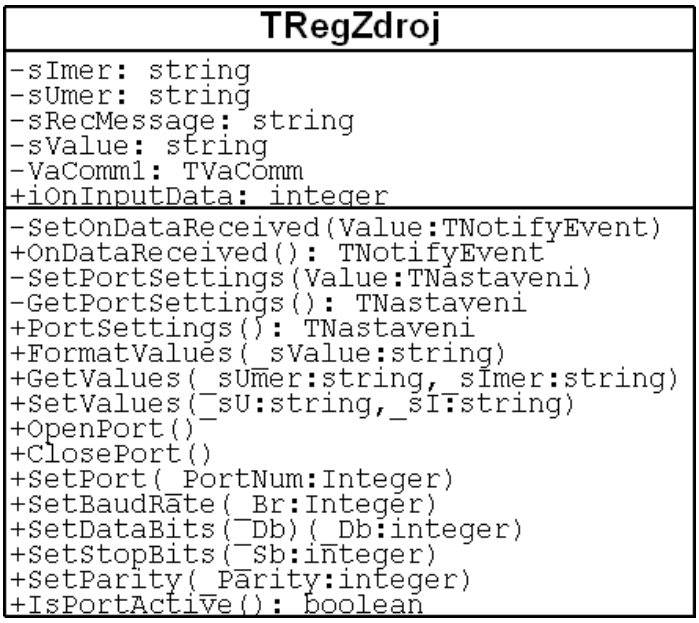
Obrázek 14: Grafické znázornění částí programu



Obrázek 15: Hierarchie programu

5.1 Popis třídy TRegZdroj

Třída TRegZdroj (obrázek 16) reprezentuje logiku ovládání zdroje. V atributech `sImer` a `sUmer` jsou uloženy aktuální měřené hodnoty na výstupu zdroje. Do atributu `sRecMessage` jsou ukládána data z příchozího bufferu. Atribut `sValue` obsahuje napětí a proud, které je odesláno regulovatelnému zdroji. Pomocí `VaComm1` přistupujeme k metodám třídy `TVaComm`. `OnInputData` nese informaci zda přicházejí data od regulovatelného zdroje.



Obrázek 16: Popis třídy TRegZdroj

Metody Třídy TRegZdroj

`SetOnDataReceived(Value:TNotifyEvent)`

Přiřazení adresy metody obsluhy po přihlášení k odběru událostí.

`OnDataReceived():TNotifyEvent`

Vlastní událost, která je vyvolána při zpracování přijatých dat.

`PortSettings():TNastaveni`

Property, která umožňuje hromadné nastavení a získání parametrů přenosu a čísla sériového portu. Předávan je typ TNastaveni (viz. unita URegZdroj). Pokud chceme získat nastavení ser. portu `tmpNast:=RegZdroj.PortSettings;` je vyvolána metoda `GetPortSettings()` a v případě změny nastavení portu

`RegZdroj.PortSettings:=tmpNast;` je volána metoda `SetPortSettigs()`. `GetPortSettings()` a `SetPortSettigs()` nastavení port buď získají nebo zapíší.

`FormatValues(_sValue:string)`

Naformátuje zpracovaný řetězec a data nastaví do atributů `sUmer` a `sImer`.

`GetValues(var _sUmer,sImer:string)`

Vrátí aktuální hodnoty na výstupu zdroje.

`SetValues(_sU,sI:string)`

Nastaví hodnoty na výstupu zdroje. Hodnoty jsou předávany ve formátu uuu a iii. (Např. `_sU:=123` znamená napětí 12,3 V `_sI:=123` znamená proud 1,23 A).

`OpenPort()`

Otevře sériový port.

`ClosePort()`

Zavře sériový port.

`SetPort(_PortNum:integer)`

Nastaví číslo sériového portu.

`SetBaudRate(_Br:integer)`

Nastaví přenosovou rychlost.

`SetDataBits(_DB:integer)`

Nastaví počet datových bitů při přenosu.

`SetStopBits(_SB:integer)`

Nastaví počet stop bitů při přenosu.

`SetParity(_Parity:integer)`

Nastaví přítomnost, případně druh parity.

`IsPortActive:boolean`

Vrací true když je port otevřen, false, když je zavřen.

5.2 Komunikační protokol

Regulovatelný zdroj komunikuje v ASCII režimu. Veškerá komunikace tedy probíhá znakově. Pro nastavení výstupních parametrů (napětí a proudu) je nutné odeslat obě hodnoty po sobě ve tvaru:

`uuu↵iii↵`.

Po odeslání každé z hodnot musí následovat nečíselný znak. (Například návrat vozíku (carriage return), který má ordinání hodnotu #13. Odchozí zpráva může vypadat následovně: `101↵123↵`. Tímto způsobem je zdroji odesláno, aby nastavil napětí na 10,1 V a proud na 1,23 A.

Příchozí data jsou formátována ve tvaru:

`Duuuuuiiiii↵`.

Po D následuje pět číselných znaků pro napětí a pět pro proud. První číslice (z pětice) je vždy nulová. Poslední číslice z první pětice představuje napětí 10 mV. V případě proudu je první číslice opět vždy nulová a poslední představuje proud 1 mA. Příchozí paket tedy může vypadat následovně: `D0251201252`. V tomto případě zdroj odeslal informaci, že napětí na výstupu je 25,12 V. Proud na výstupu je 1,252 A.

V hlavním programu se zobrazuje napětí řádově na desetiny V a proud řádově na setiny A s ohledem na možnosti nastavení hodnot na zdroji, které jsou ve zmiňovaném rozsahu.

5.3 Zpracování příchozích dat pomocí událostí

Třída `TRegZdroj` zapouzdřuje komponentu `TVaComm`. Při spuštění programu se vytvoří instance třídy `TRegZdroj`, která reaguje na událost `OnRxChar`() (příchozí paket) třídy `TVaComm`. Při vyvolání události `OnRxChar`() jsou přijatá data zpracována třídou `TRegZdroj` a následně tato třída vyvolá událost `OnDataReceived()`. V hlavním programu je nutné se přihlásit k odběru událostí. Při vzniku události `OnDataReceived()` jsou v aplikaci zobrazeny aktuální hodnoty na výstupu regulovatelného zdroje.

5.4 Popis programu

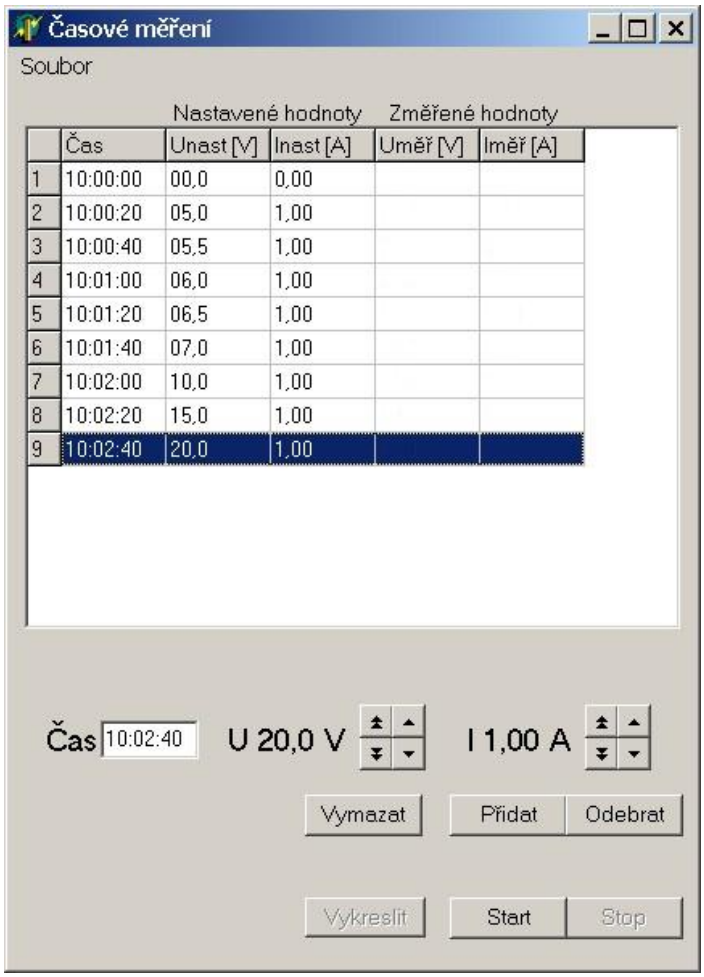
V hlavním okně programu (obrázek 17) se zobrazují nástavené a naměřené hodnoty. Po stisknutí tlačítka ON se zahájí komunikace přes sériový port. Pokud dojde k chybnému nastavení sériového portu, port se neotevře a uživatel je vyzván ke změně nastavení. Výstupní hodnoty se nastavují pomocí dvou čtveřic tlačítek stejným způsobem jako na regulovatelném zdroji (kapitola 1.1).

Obrázek 17: Hlavní okno řídící aplikace

5.5 Nastavení výstupních hodnot v čase

Pomocí této funkce programu je možné automatizovaně provádět jednoduchá měření elektronických součástek. Prostřednictvím formuláře (viz. obrázek 18) lze nastavit požadované napětí a proud na výstupu zdroje na konkrétní hodinu, minutu a vteřinu. V jedné měřící úloze lze provést 100 měření. Po nastavení napětí a proudu na výstupu se hodnoty zaznamenají do formuláře. Měřená data je možné exportovat do souboru CSV (comma separated value). V tomto formátu jsou jednotlivé hodnoty odděleny středníkem. Program umožňuje tyto soubory ukládat i načítat. Pokud je uživatelem vytvořen vlastní CSV soubor se vstupními daty, pak je nutné přesně dodržet formát vstupních dat, jinak měření neproběhne korektně. Formát jednoho řádku vstupních dat je následující: 1;10:22:30;12,3;0,09 (číslo měření;čas hh:mm:ss; uu,u [V]; i,ii [A]) – vzorový soubor je k dispozici v adresáři programu). Pro správné provedení časového

měření je nutné nastavit odstup minimálně 20 vteřin mezi jednotlivými změnami hodnot. Tato doba je potřebná pro změnu nastavení z předchozí hodnoty na hodnotu novou. Po korektně provedeném měření (změřené hodnoty se zobrazí ve sloupcích Uměř a Iměř) je možno vykreslit graf závislosti napětí na proudu – voltampérovou charakteristiku. Po exportu dat do CSV souboru je možné následné zpracování měřených dat (například v aplikaci MS Excel).



Obrázek 18: Nastavení výstupních hodnot v čase

6 Závěr

Bakalářská práce navazuje na moji závěrečnou práci na Střední průmyslové školy elektrotechnické v Pardubicích, kde jsem si zvolil jako téma digitálně regulovatelný laboratorní zdroj.
Tento maturitní výrobek jsem zkonstruoval po elektrotechnické i mechanické stránce. V rámci této práce jsem rozšířil jeho ovládání prostřednictvím PC a rozhraní RS 232.
Vyřešil jsem komunikaci tohoto zdroje s PC pomocí aplikace ve vývojovém prostředí Delphi pracujícím pod OS Windows.
Pro oddělení grafického návrhu programu od ovládání zdroje, v rámci zásad objektově orientovaného programování, jsem uplatnil princip tzv. událostí, který je charakteristický pro programování v Delphi. Zdrojové kódy programu jsou uvedeny v příloze na CD.

V této práci jsem uvedl veškeré potřebné informace od popisu principů komunikace po RS232, volby vhodné komponenty pro vlastní komunikaci, přes její instalaci, popis a praktické použití.

Výsledkem práce je řídící aplikace, která umožňuje nastavení a čtení měřených hodnot digitálně regulovatelným laboratorním zdrojem. Výstupní parametry je možno nastavovat v čase. Pomocí tohoto nástroje je možné automatizovat jednoduché měřící úlohy, například změřit voltampérovou charakteristiku elektronických součástek.

Seznam použité literatury

[1] Gook, M. *Hardwarová rozhraní: průvodce programátora*. Brno: Computer Press, 2006. 463 s. ISBN 80-251-1019-2

[2] Matoušek, D. *Udělejte si z PC v Delphi elektronickou laboratoř a řídící centrum ve Windows 1. díl.* Praha : BEN, 2003. 271 s. ISBN 80-7300-111-X

[3] Messmer, H.-P., Dembowski, K. *Velká kniha hardware: architektura, funkce, programování.* Brno: Computer Press, 2005. 1224 s. ISBN 80-251-0416-8

[4] Sedláček, J., Slaba, J. *Delphi v kostce.* Praha : BEN, 2000. 513 s. ISBN 80-86056-97-X

[5] Svoboda, L., Voneš, P., Konšal, T. *1001 tipů a triků pro Delphi*. Brno : Computer Press 2003. 546 s. ISBN 80-7226-488-5.

[6] Vlach, J. *Počítačová rozhraní: Přenos dat a řídící systémy*. Praha : BEN, 2001. 175 s. ISBN 80-7300-010-5

[7] Vykopal, J. *Sériové rozhraní v Delphi*. [online]. 20.11. 2001. Dostupné z WWW <http://www.builder.cz/serial85.html>.

Přílohy

[1] Řídící aplikace k digitálně regulovatelnému laboratornímu zdroji na CD v adresáři Řídící aplikace.

[2] Dokumentace k digitálně regulovatelnému laboratornímu zdroji na CD v adresáři RegZdroj.

ÚDAJE PRO KNIHOVNICKOU DATABÁZI

Název práce	Ovládací program k regulovatelnému laboratornímu zdroji
Autor práce	Petr Křivka
Obor	Informační technologie
Rok obhajoby	2007
Vedoucí práce	Ing. Martin Hájek
Anotace	Teoretická část bakalářské práce popisuje principy komunikace po sériovém rozhraní RS232, principy programování RS232 pod OS Windows, dostupné komponenty pro vývojové prostředí Borland Delphi a využití událostmi řízeného programovaní pro vývoj řídících aplikací. Aplikační část se zabývá implementací vybrané komponenty pro komunikaci mezi RS232 a konkrétním digitálně regulovatelným zdrojem.
Klíčová slova	RS232, sériový port, COM port, komunikace, Delphi, události, Windows