VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DIGITÁLNÍ STOPKY PRO DVA ČASY

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE PETER ŠILON
AUTHOR

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DIGITÁLNÍ STOPKY PRO DVA ČASY

DIGITAL STOPWATCHES WITH TWO TIMES

BAKALÁŘSKÁ PRÁCE

BACHELOR´S THESIS

AUTOR PRÁCE PETER ŠILON
AUTHOR

VEDOUCÍ PRÁCE ING. TOMÁŠ HERRMAN
SUPERVISOR

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Šilon Peter**
Obor: Informační technologie
Téma: **Digitální stopky pro 2 časy**
Kategorie: Uživatelská rozhraní

Pokyny:

1. Navrhněte vhodný způsob připojení dvou startovacích a dvou zastavovacích kontaktů k PC.
2. Seznamte se s ovlávádním zvoleného rozhraní v prostředí Windows.
3. Seznamte se s multithredingem v prostředí Windows.
4. Implementujte aplikaci, která umožní měření dvou časů pomocí externího startovacího a zastavovacího pulzu.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

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

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

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

Vedoucí: **Herrman Tomáš, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Peter Šilon**
Id studenta: 84072
Bytem: Družstevná 196, 059 42 Gerlachov
Narozen: 10. 03. 1985, Poprad
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   bakalářská práce

Název VŠKP: Digitální stopky pro 2 časy
Vedoucí/školitel VŠKP: Herrman Tomáš, Ing.
Ústav: Ústav počítačových systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☒ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Cieľom bakalárskej práce bolo navrhnúť vhodné pripojenie dvoch štartovacích a dvoch zastavovacích kontaktov k PC. Práca si kladie za cieľ oboznámiť čitateľa s možnosťami ovládania portov pod *MS Windows*. Jedná sa o popis špecifických funkcií *WinAPI* a riešenie otázky priameho prístupu na porty. V ďalších kapitolách je rozoberaná problematika presného merania času na aplikačnej úrovni a oboznámenie sa s tvorbou multivláknových aplikácii v prostredí *MS Windows*. Druhá časť bakalárskej práce sa venuje konkrétnej implementácii digitálnych stopiek na báze mikroprocesoru.

Kľúčové slová

presné meranie času, sériový port, priamy prístup na porty, WinAPI, multithreading, mikroprocesory rady 8051

Abstract

The aim of my bachelor thesis is to suggest the appropriate connection of two starting and two stopping contacts to PC. The study is going to inform its readers about the opinions how to control the *MS Windows* ports. It concerns about the specific description of *WinAPI* functions as well as the solution for the direct accesion to the ports. Next chapters point at the issue of accurate time measurement on the application level and at the familiarisation with the proces of creation of the multithreading applications in *MS Windows*. The second part of bachelor thesis focus on extensional implementation of stopwatches based on microcontroller.

.

Keywords

exact time measurement, serial port, direct access to the ports, WinAPI, multithreading, 8051 architecture microcontrollers

Citácie

Peter Šilon: Digitálne stopky pre dva časy, Brno, FIT VUT v Brne, 2007

Digitálne stopky pre dva časy

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Tomáša Herrmana.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Peter Šilon, 4.5.2007

Poďakovanie

Rád by som poďakoval všetkým, ktorí mi pomohli pri realizácii mojej bakalárskej práce. Menovite Ing. Tomášovi Herrmanovi za odbornú pomoc a Ing. Pavlovi Muchovi za pomoc pri výrobe plošného spoja.

© Peter Šilon, 2007.
Táto práca vznikla ako školské dielo na Vysokom učení technickom v Brne, Fakulte informačných technológií. Práca je chránená autorským zákonom a jej použitie bez udelenia oprávnenia autorom je nezákonné, s výnimkou zákonom definovaných prípadov.

Obsah

1 ŠTANDARDNÉ ROZHRANIA PC 4

1.1 SÉRIOVÝ PORT 4

1.1.1 Prevodníky RS232 - TTL 5

1.1.2 Parametre dátového prenosu 6

1.1.3 Riadenie toku 6

1.1.4 Synchrónny a asynchrónny prenos 6

1.2 PARALELNÝ PORT 8

1.2.1 SPP 8

1.2.2 EPP 9

1.2.3 ECP 9

1.3 USB 9

2 OVLÁDANIE PORTOV V OPERAČNOM SYSTÉME WINDOWS 10

2.1 PRIAMY PRÍSTUP NA PORTY 10

2.1.1 Riešenie 11

2.2 WINDOWS API A PORTY 12

2.2.1 Otvorenie portu 12

2.2.2 Zápis na port 13

2.2.3 Čítanie z portu 13

2.2.4 DCB štruktúra 14

2.2.5 Timeouty 14

2.2.6 Zatvorenie portu 15

3 MULTITHREADING V PROSTREDÍ WINDOWS 15

4 MERANIE ČASU V PROSTREDÍ WINDOWS 16

4.1 ČASOVAČE 16

4.2 ČÍTAČE 17

5 STRUČNÝ POPIS MIKROPROCESORU 2051 18

6 NÁVRH 19

6.1 POŽIADAVKY 19

6.2 MOŽNOSTI 19

7 HARDWAROVÁ REALIZÁCIA 20

7.1 POPIS ZAPOJENIA 20

7.2 PRIPOJENIE KONTAKTOV 21

7.3 POPIS FUNKCIE 22

7.4 KONŠTRUKCIA DPS 22

8 SOFTWAROVÁ REALIZÁCIA 22

8.1 FIRMWARE 22

8.1.1 Meranie času 24

8.1.2 Ukladanie času 25

8.2 KOMUNIKAČNÝ PROTOKOL 26

8.2.1 Zabezpečenie komunikácie 27

8.2.2 Prístup na zbernicu 28

8.3 UŽÍVATEĽSKÉ ROZHRANIE 28

8.3.1 Implementácia 30

9 POROVNANIE 32

10 ZÁVER 33

Úvod

Čas je základnou fyzikálnou veličinou, ktorá vyjadruje interval medzi dvoma udalosťami alebo trvanie deja. O meranie času sa ľudia usilujú od nepamäti. K prvotnému meraniu času sa využíval pravidelný pohyb Slnka, Mesiaca a hviezd. Najznámejším prístrojom využívajúcim tieto javy sú slnečné hodiny. Spolu s rozvojom priemyslu sa rozvíjali aj metódy merania času a začali sa objavovať prvé mechanické hodiny. Ich základom je pravidelný pohyb kyvadla, ktoré určuje rýchlosť chodu hodín. Zďaleka najrozšírenejším spôsobom ako merať čas, sa stalo až využitie piezoelektrického kryštálu. Vďaka relatívne dobrej presnosti, nízkym výrobným nákladom a jednoduchým prepojením s elektronickými obvodmi, je piezoelektrický kryštál základom mnohých zariadení pracujúcich s časom. Výhody digitálneho spracovania údajov viedli k myšlienke merania času na počítači. V nasledujúcom dokumente je preto rozoberaná problematika zostavenia digitálnych stopiek pripojiteľných k počítaču.

Prvá kapitola sa zaoberá popisom štandardných rozhraní počítača, sériového, paralelného a USB portu. Druhá kapitola popisuje ovládanie spomínaných portov v operačnom systéme Windows a rieši problematiku priameho prístupu na porty. Tretia kapitola je stručný úvod k vytváraniu multivláknových aplikácií v prostredí Windows. Vo štvrtej kapitole sa čitateľ oboznámi s možnosťami merania času na aplikačnej úrovni pomocou systémových čítačov a časovačov. Piata kapitola uvádza stručný popis mikroprocesoru *ATMEL 89c2051*, ktorý je použitý pri návrhu digitálnych stopiek. V šiestej kapitole sú uvedené požiadavky a možnosti, z ktorých som pri návrhu vychádzal. Siedma a ôsma kapitola sú už venované konkrétnej implementácii stopiek, hardwarového aj softwarového vybavenia. Záver dokumentu hodnotí dosiahnuté výsledky a mieru, akou sa podarilo splniť požiadavky zadania.

1 Štandardné rozhrania PC

Ak chceme pomocou počítača riadiť alebo sledovať reálne procesy, potrebujeme v prvom rade vhodné spojenie počítača a okolitého sveta. Program musí mať možnosť prijímať informácie zvonku a vysielať riadiace informácie pripojeným zariadeniam. Potrebnú bránu k vonkajšiemu svetu predstavuje rozhranie počítača (interface).

1.1 Sériový port

Veľká väčšina počítačov disponuje jedným alebo dvomi sériovými portami, zvyčajne označenými ako *COM1*, *COM2*, atd. Jeho pôvodným účelom bolo spojenie medzi počítačom a modemom, aby bolo možné prenášať dáta po telefónnej linke. Často sa však na sériový port pripájali aj iné zariadenia ako myš alebo tlačiareň.

Výhody použitia sériového portu:

- Rozhranie je veľmi odolné voči zničeniu
- Zariadenia je možné pripojiť/odpojiť pri zapnutom počítači
- Napájacie napätie pre jednoduché zariadenia je možné odoberať priamo z portu
- Výstupy sú odolné voči skratu
- Výstupy môžu dodávať prúd až 10mA, takže môžu budiť LED alebo výkonové stupne

Pin	***I/O***	***Označenie***	***Funkcia***
1	IN	DCD	Detektor prijímaného signálu
2	IN	RxD	Prijímané dáta
3	OUT	TxD	Vysielané dáta
4	OUT	DTR	Pohotovosť DTE
5	---	GND	Signálová zem
6	IN	DSR	Pohotovosť DCE
7	OUT	RTS	Výzva k vyslaniu
8	IN	CTS	Pohotovosť k vyslaniu
9	IN	RI	Indikátor volania

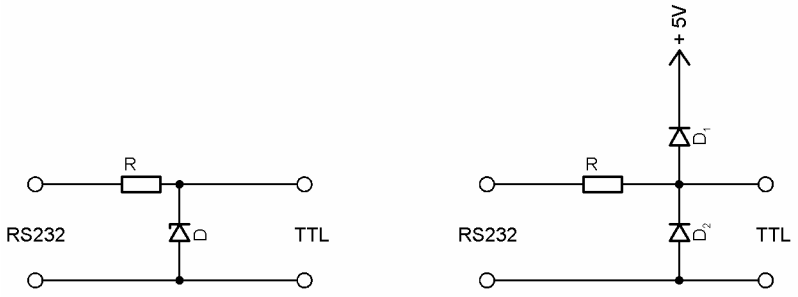
Tabuľka 1.1: Zapojenie 9 pinového konektoru

Vlastný prenos dát na sériovom rozhraní sa uskutočňuje po linkách *TxD* (Transmit Data) a *RxD* (Receive Data). Všetky ostatné linky plnia pomocné funkcie pri riadení prenosu. Obyčajne sa označujú výrazom „handshake“. Tieto linky môžme programovo nastaviť alebo čítať ich stav.

Elektrické vlastnosti vstupov a výstupov sú dané normou RS232: v stave nízkej úrovne (L) majú výstupy napätie 15V, naopak v stave vysokej úrovne (H) je to napätie -15V. Vstupy majú vstupný odpor približne 10kΩ a napätie menšie ako -3.0V rozpoznávajú ako vysoké (H), kým napätie väčšie ako 3.0V rozpoznávajú ako nízke (L). Vstupné hodnoty sa teda rozpoznávajú, iba ak ležia mimo interval (-3.0V; 3.0V), a je preto možné použiť TTL logiku.

1.1.1 Prevodníky RS232 - TTL

Pri použití sériového portu na ovládanie alebo riadenie určitého zariadenia je potrebné zaistiť prevod úrovní RS232 na úrovne používané zariadením, inak by mohlo dôjsť k poškodeniu vstupných obvodov zariadenia. Najčastejšie ide o prevod na úroveň TTL logiky. Na *obr.1.1a* a *1.1b* sú zobrazené jednoduché prevodníky využívajúce diódy.



Obr. 1.1a *Obr. 1.1b*

Zapojenie na *obr.1.1a* využíva pracovný odpor a Zenerovu diódu. Záverné napätie diódy volíme 4.7V. Ak je vstupné napätie záporné, otvorí sa dióda v priepustnom smere a na výstupe bude napätie -0.7V. Ak je vstupné napätie kladné a väčšie ako zvolená hodnota 4.7V, bude obmedzené práve na tejto hodnote. Druhé zapojenie je o niečo komplikovanejšie. Ak je vstupné napätie záporné, otvorí sa dióda D2 v priepustnom smere a výstupné napätie bude -0.7V. Pokiaľ je na vstupe napätie vyššie ako 5.7V, otvorí sa dióda D1 a na výstupe sa objaví napätie práve 5.7V.

Najkomplexnejšiu variantu prevodu úrovní ponúka integrovaný prevodník *MAX232* od firmy Maxim Integrated Circuits. Obsahuje dva nezávisle kanály pre prevod RS232 na TLL a naopak. Obvod má pre svoju činnosť zabudovaný násobič napätia a invertor. Oba kanály signál invertujú, aby sa dosiahlo požadovaného výstupu. Násobič aj invertor potrebujú k svojej činnosti pripojiť externé kondenzátory. Výnimkou je obvod *MAX233*, ktorý má kondenzátory integrované.

1.1.2 Parametre dátového prenosu

Najjednoduchší spôsob, ako bez nárokov na výkon počítača zabezpečiť prenos dát, je parita. Vo vysielacom zariadení sa sčíta počet jednotkových bitov a doplní sa paritný bit tak, aby ostala zachovaná podmienka párneho alebo nepárneho počtu jednotkových bitov.

- **Párna parita** – počet jednotkových bitov + parita = párne číslo
- **Nepárna parita** – počet jednotkových bitov + parita = nepárne číslo
- **Space parita** – paritný bit je stále nulový, používa sa napríklad pri komunikácii 7 - bitového zariadenia s 8 - bitovým.
- **Mark parita** – paritný bit je trvalo nastavený na jedna.

Ďalším parametrom je počet bitov obsiahnutých v jednom byte prenášaných dát. 7 – bitový formát sa niekedy používal na termináloch kvôli ušetreniu jedného bitu. Už sa prakticky nepoužíva ale stal sa štandardom. Dnes sa používa výhradne 8 – bitový formát.

Stop bit definuje ukončenie rámca. Zároveň definuje určitý časový odstup pre prijímač. Práve počas príjmu stop bitu väčšina zariadení spracúva prijatý byte. Pri pomalších zariadeniach sa stop bit zdvojuje, aby príjmač stihol spracovať prijatý byte. Jedná sa o štandard 110 Baud.

1.1.3 Riadenie toku

Riadenie toku dát (handshaking) predstavuje potvrdenie príjmu alebo pripravenosť k prenosu a jeho zahájeniu na úrovni softwarového alebo hardwarového rozhrania.

Pri hardwarovom riadení na jednej strane vysielač vyšle signál o pripravenosti platných dát, na strane druhej prijímač vyšle signál, že je schopný dáta spracovať.

Softwarový handshaking prebieha na úrovni komunikačných protokolov. Pomocou bežného dátového kanálu prijímač vysielaču oznámi, či je schopný dáta prijať a spracovať. BIOS v PC používa pre softwarový handshaking znaky XON/XOFF z ASCII tabuľky. Ak je potrebné uvedené znaky použiť v kontexte správy, ich výskyt sa zdvojuje, čo prenos samozrejme o niečo spomalí.

1.1.4 Synchrónny a asynchrónny prenos

Synchrónny prenos informácií znamená, že sa na nejakom vodiči alebo vodičoch nastaví určitá úroveň, ktorá prenáša informáciu a validita sa potvrdí impulzom alebo zmenou úrovne synchronizačného signálu. Z toho vyplýva, že k vlastnému prenosu potrebujeme jeden signálový vodič navyše.

Základné vlastnosti synchrónneho prenosu:

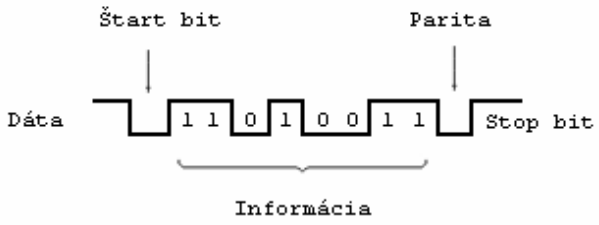
- Vhodné pre veľké objemy dát prenášané po viacerých vodičoch
- Nutné určiť, kto vysiela synchronizačné impulzy
- Možno použiť premenlivú rýchlosť prenosu
- Synchronizačný vodič navyše
- Jednoduchá elektronika na prijímacej strane

Asynchrónny prenos dát prenáša dáta v určitých sekvenciách. Dáta sú prenášané presne definovanou rýchlosťou. Na začiatku sa zašle štartovacia sekvencia, na ktorú sa synchronizujú všetky prijímacie zariadenia. Všetky strany obsahujú presný oscilátor, vďaka ktorému prečítajú dáta v presne definovaných intervaloch. Po ukončení sekvencie je ďalší príjem znova synchronizovaný štartovacou sekvenciou.

Základné vlastnosti asynchrónneho prenosu:

- Nevhodné pre veľké objemy dát
- Vhodné na väčšie vzdialenosti
- Použiteľné pri komunikácii medzi viacerými zariadeniami
- Jednoznačne definovaná prenosová rýchlosť
- Zložitá elektronika obvodov
- Vzhľadom na štart, stop a paritný bit je rýchlosť prenosu užitočných informácií voči synchrónnemu prenosu menšia zhruba o 20%

RS232 používa asynchrónny prenos. Každý prenesený byte treba synchronizovať. Na to sa využíva zostupná hrana štart bitu za ktorou nasledujú prenášané dáta.



Obr. 1.2: Asynchrónny prenos

1.2 Paralelný port

Paralelný port *LPT* je štandardnou súčasťou počítačov PC pre paralelnú komunikáciu s periférami pomocou 17 digitálnych liniek, ktoré môžeme rozdeliť na 8 dátových a 9 riadiacich liniek. PC je možné s periférami spoľahlivo prepojiť na vzdialenosť dvoch metrov. Pôvodne bol paralelný port vytvorený pre komunikáciu s tlačiarňou, teda jednosmerný prenos dát z PC do tlačiarne. Neskôr bol však štandardizovaný podľa *IEEE 1284*. Tento štandard definuje 3 základné módy činnosti paralelného portu.

- **SPP** – Centronics mode
- **EPP** – Enhanced Paralel Port
- **ECP** – Extended Capabilities Mode

Pin	***I/O***	***Označenie***	***Register***
1	IN/OUT	nStrobe	Control
2	OUT	Data 0	Data
3	OUT	Data 1	Data
4	OUT	Data 2	Data
5	OUT	Data 3	Data
6	OUT	Data 4	Data
7	OUT	Data 5	Data
8	OUT	Data 6	Data
9	OUT	Data 7	Data
10	IN	nACK	Status
11	IN	BUSY	Status
12	IN	Paper END	Status
13	IN	Select	Status
14	IN/OUT	Linefeed	Control
15	IN	Error	Status
16	IN/OUT	INIT	Status
17	IN/OUT	Select Printer	Control
18 - 25	--	GND	--

Tabuľka 1.2: Zapojenie 25 pinového konektoru

1.2.1 SPP

Tento mód je tiež označovaný ako *Centronics mode* a je to základný mód pre štandardný paralelný port, určený pre jednosmernú komunikáciu s tlačiarňou. Definovaný je prenos typu forward, teda prenos dát smerom z PC k periférii a rýchlosť prenosu sa pohybuje len okolo hodnoty 150 KB/s.

SPP mód je riadený tromi základnými registrami. Dátový register (offset 1), určený pre zápis vysielaných dát, stavový register (offset 2), určený pre získanie stavov na linke a riadiaci register (offser 3), určený pre riadenie pripojenej periférie.

1.2.2 EPP

Enhanced Paralel Port protokol bol pôvodne vyvinutý firmami *Intel*, *Xircom* a *Data Systems*, aby poskytol výkonné prepojenie cez paralelný port a bol pritom kompatibilný so štandardným typom.

Efekt tohto módu je prenosová rýchlosť od 500 KB/s do 2MB/s. Rýchlosť prenosu je daná rýchlosťou najpomalšieho komunikujúceho zariadenia. To si komunikujúce strany zistia výmenou signálov *nWait* a *nDataStrobe*.

1.2.3 ECP

Extended Capability Port protocol bol vytvorený firmami *HP* a *Microsoft* ako pokročilý mód pre obojsmernú komunikáciu PC a ďalších periférií ako napríklad skener. Prenosová rýchlosť ECP je 800 KB/s bez využitia DMA a až 2MB/s s využitím DMA. Toto platí pre zbernicu ISA. Pre zbernicu PCI sú hodnoty 3 až 5 MB/s.

1.3 USB

Je univerzálna sériová zbernica (Universal Serial Bus). Predstavuje moderný spôsob pripojenia širokého spektra periférií k počítaču PC. Prenosové rýchlosti sa pohybujú v rozmedzí od 1 až do 480 Mb/s. Výhodou je možnosť napájať periférie priamo z portu USB. Rozhranie obsahuje 5V napätie a umožňuje odber prúdu max. 100mA. K jednému USB portu počítača je možné pripojiť až 127 rôznych zariadení.

2 Ovládanie portov v operačnom systéme Windows

Funkcie *Windows API* (skrátene *Win API*) tvoria jednotné rozhranie pre vývoj aplikácií pre operačný systém Windows. Win API poskytuje priamu podporu iba pre sériové a paralelné porty. Táto podpora sa v oboch prípadoch výrazne odlišuje.

Paralelný port je podporovaný iba v móde SPP. Sériový port je podporovaný desiatkami funkcií, ktoré umožňujú veľmi pohodlné nastavenie sériového rozhrania a monitorovanie jeho stavu.

Vo väčšine prípadov si teda vystačíme s funkciami Win API (hlavne pri sériovom porte). Niekedy však bude potrebný priamy prístup na porty, ktorý nie je funkciami Win API podporovaný.

2.1 Priamy prístup na porty

V niektorých prípadoch nejde používať funkcie Win API, pretože to povaha úlohy nedovoľuje. Jedná sa napríklad o riadenie zásuvných kariet alebo netradičné použitie niektorých periférií zabudovaných v počítači (napr. časovač).

Problém priameho prístupu na porty počítača vzniká pri používaní operačných systémov založených na jadre NT, teda *Windows NT/2000/XP*, ale svoj pôvod má niekde úplne inde. Môže za to chránený režim činnosti procesorov Intel, ktorý tieto systémy využívajú. Ten totiž sám vykonáva operácie spojené s využívaním portov, adresovaním pamäti a pod. Chránený režim poskytuje štyri rozličné úrovne prístupu, ktoré sa označujú ring 0, ring 1, ring 2 a ring 3. Ring 0 je najvyššia úroveň oprávnenia, ring 3 naopak najnižšia. Operačné systémy Windows používajú len ring 0 alebo 3. Ak je kód vykonávaný v režime ring 0, môže obsahovať ľubovoľné inštrukcie, teda aj inštrukcie prístupu k vstup/výstupným portom. Pod Windows beží v ring 0 iba jadro operačného systému a ovládače zariadení. Všetky užívateľské aplikácie bežia v ring 3, preto sú ich možnosti prístupu na porty obmedzené. Procesory Intel 386 a vyššie mapujú každý port ring 3 programu pomocou *IOPM* (Input/Output Privilege Mask), tzv. povoľovacej masky. Každý z bitov tejto masky odpovedá jednej vstup/výstupnej adrese. Keď sa ring 3 program pokúsi vykonať inštrukciu IN alebo OUT, testuje procesor *IOPM* masku, aby určil či je možné k portu pristúpiť. Pokiaľ je programu prístup zamietnutý, generuje procesor výnimku, ktorú zachytí operačný systém, a ten nedovolí daný proces dokončiť. V spracovaní tejto výnimky sa operačné systémy odlišujú.

- **Windows 95/98/ME** – reagujú povolením prístupových práv úpravou *IOPM* masky. Prvá inštrukcia IN/OUT vyvolá výnimku, ktorá spôsobí, že sa upraví príslušný bit masky. Nasledujúce inštrukcie už výnimku nevyvolajú.
- **Windows NT/2000/XP** – tieto systémy sú veľmi striktné, pretože sa jedná o robustné systémy so zvýšenou bezpečnosťou. Ak je operačným systémom zachytená výnimka o prístup k portom, je program prerušený a inštrukcia nie je dokončená.

2.1.1 Riešenie

Ak chceme aj napriek uvedenému problému získať priamy prístup k portom, máme len dve možnosti:

- Napísať vlastný ovládač, ktorý bude bežať v režime ring 0 a prenášať dáta medzi týmto ovládačom a ring 3 programom
- Zmeniť *IOPM* masku ring 3 programu, čo by viedlo k povoleniu vstup/výstupných operácií. Táto možnosť nie je doporučená, lebo vedie k zníženiu bezpečnosti celého systému.

V prvom spomínanom prípade ide o vytvorenie špeciálneho ovládača pracujúceho v privilegovanom režime ring 0. Dáta medzi ovládačom a aplikačným programom je možné prenášať pomocou systémových volaní *IOCTL*. Je potrebné mať na pamäti, že každý prístup na porty má za následok prepnutie procesoru z režimu ring 3 do ring 0, vykonanie operácie a opätovné prepnutie procesoru do ring 3. Pre novo vytvárané programy je tento spôsob výhodnejší.

Druhý spôsob sa hodí pre aplikácie vytvorené pod *Windows 9x*, ktoré chceme bez úprav spúšťať vo *Windows NT*. Aj v tomto prípade potrebujeme ovládač bežiaci v privilegovanom móde. Ten podľa identifikátora spúšťaného procesu upraví príslušný bit povoľovacej masky a aplikácii tak umožní priamy prístup na porty bez nutnosti ďalších systémových volaní.

Na internete sú dostupné viaceré univerzálne ovládače, ktoré riešia priamy prístup k portom. Jedným z nich je ovládač *User Port SYS*, ktorý vytvára podmienky pre všetky aplikácie spúšťané pod *Windows NT* a vyžadujúce priamy prístup k portom. Nevýhodou je možnosť viacnásobného prístupu k jednému portu, čo vedie k nemalým nepríjemnostiam. *Port Talk* je ďalším z univerzálnych ovládačov, ktorý umožňuje programátorovi rovno vytvárať aplikácie s priamym prístupom na porty.

2.2 Windows API a porty

Nasledujúca kapitola približuje základné funkcie Win API pre prácu s portami.

2.2.1 Otvorenie portu

Na otvorenie komunikačného portu slúži funkcia `CreateFile`. Treba si uvedomiť, že paralelný a sériový port sú v pojatí operačného systému Windows chápané ako súbory s menami *LPT1*, *COM1*, *COM2*, atd.

```
HANDLE CreateFile (
      LPCTSTR lpFileName,
      DWORD dwDesiredAccess,
      DWORD dwShareMode,
      LPSECURITY_ATTRIBUTES lpSecAttributes,
      DWORD dwCreationDisposition,
      DWORD dwFlagsAndAttributes,
      HANDLE hTemplateFile
);
```

- **lpFileName** – obsahuje názov portu („COM1“, „LPT1“, ...)
- **dwDesiredAccess** – určuje požadovaný prístup k portu. Obvykle hodnota GENERIC_READ | GENERIC_WRITE
- **dwShareMode** – pre prípad portov sa nastaví na NULL (porty nejde zdieľať)
- **lpSecAttributes –** hodnota NULL predstavuje východzie zabezpečenie
- **dwCreationDisposition** – chovanie pri otváraní alebo vytváraní súboru. Pre porty sa nastaví na OPEN_EXISTING (otvorí existujúci port).
- **hTemplateFile** – súbor, ktorý sa použije ako šablóna

Parameter *dwFlagsAndAttributes* definuje príznaky pre otváraný port. Hodnota 0 znamená, že portu nedefinujeme žiadne príznaky a otvorí sa v tzv. *nonoverlapped* režime. Druhá možná hodnota, *FILE_FLAG_OVERLAPPED*, spôsobí otvorenie portu v *overlapped* režime.

Port v nonoverlapped režime má určité obmedzenia. Pokiaľ sa vstup/výstupná operácia vykonáva, volajúce vlákno je zablokované. Až keď je operácia vykonaná, vlákno môže pokračovať ďalej. Tento režim je vhodný pre multivláknové aplikácie, pretože ak je jedno vlákno zablokované vstup/výstupnou operáciou, ostatné vlákna môžu pokračovať v činnosti. Je však na zodpovednosti aplikácie, aby synchronizovala prístup k portom.

Overlapped režim je o niečo komplikovanejší, ale umožňuje väčšiu flexibilitu a efektívnosť. Port otvorený v overlapped režime umožňuje viacerým vláknam vykonávať vstup/výstupné operácie súčasne. Mimo to, chovanie overlapped operácií dovoľuje jednému vláknu volať viac rôznych vstup/výstupných funkcií a vykonávať iný kód, kým bude čakať na ich dokončenie.

2.2.2 Zápis na port

Funkcia `WriteFile` je použiteľná pre zápis dát na port. Táto funkcia je veľmi univerzálna, pretože nepracuje s konkrétnymi dátovými typmi, ale uvažuje dáta ako skupinu bytov. Dáta sú určené ich adresou v pamäti a ich dĺžkou.

```
BOOL WriteFile (
      HANDLE hFile,
      LPCVOID lpBuffer,
      DWORD numberOfBytesToWrite,
      LPDWORD lpNumberOfBytesWritten,
      LPOVERLAPPED lpOverlapped
);
```

- **hFile** – handle portu získaný funkciou `CreateFile`
- **lpBuffer** – ukazovateľ na dáta, ktoré chceme zapísať
- **numberOfBytesToWrite** – počet bytov, ktoré chceme zapísať, odpovedá veľkosti *lpBuffer*
- **lpNumberOfBytesWritten** – počet bytov, ktoré sa skutočne zapísali
- **lpOverlapped** – ukazovateľ na *OVERLAPPED* štruktúru, používa sa len pri overlapped operáciách, inak NULL

2.2.3 Čítanie z portu

Funkcia `ReadFile` je použiteľná pre čítanie z portu. Je veľmi podobná funkcii `WriteFile`, preto uvediem len jej funkčný prototyp.

```
BOOL ReadFile (
      HANDLE hFile,
      LPCVOID lpBuffer,
      DWORD numberOfBytesToRead,
      LPDWORD lpNumberOfBytesRead,
      LPOVERLAPPED lpOverlapped
);
```

2.2.4 DCB štruktúra

DCB (Device Control Block) štruktúra definuje kompletné nastavenie sériovej linky. Jednotlivé položky štruktúry zodpovedajú jednotlivým parametrom komunikácie. Pomocou funkcie `GetCommState` naplníme štruktúru aktuálnymi parametrami, funkciou `SetCommState` zasa nastavíme komunikačné parametre podľa zodpovedajúcich položiek štruktúry. Samotný zápis do štruktúry *DCB* nikde nič nezmení. Popis jednotlivých položiek štruktúry je dostupný online na [3].

2.2.5 Timeouty

Aby funkcia `ReadFile` nezostala čakať na znak, používajú sa tzv. *timeouty*, maximálne časy, ktoré má daná operácia trvať. Konkrétne hodnoty sú uložené v štruktúre `COMMTIMEOUTS`, ktorá ma tieto prvky:

- **ReadIntervalTimeout** – Maximálna doba (v ms) medzi príchodzími znakmi pri čítaní. Ak je táto doba prekročená, operácia čítania sa predčasne ukončí.
- **ReadTotalTimeoutMultiplier** – Doba (v ms), ktorá sa vynásobí počtom prijímaných bytov, a takto dá celkovú maximálnu dobu prijímania.
- **ReadTotalTimeoutConstant** – K predošlej, vypočítanej hodnote sa pripočíta ešte táto konštanta (v ms).
- **WriteTotalTimeoutMultiplier** – Doba (v ms), ktorá sa vynásobí počtom vysielaných bytov, a takto dá celkovú maximálnu dobu vysielania.
- **WriteTotalTimeoutConstant** – K predošlej, vypočítanej hodnote sa pripočíta ešte táto konštanta (v ms).

Ak je hodnota niektorého parametra 0, potom sa daný parameter neuvažuje. Po naplnení štruktúry vhodnými hodnotami sa musia stať ešte platnými pre daný komunikačný port, čo urobíme funkciou `SetCommTimeouts`.

2.2.6 Zatvorenie portu

Po skončení práce so sériovou linkou je slušné uvoľniť ju pre ďalšie programy a zatvoriť port. Funkcia očakáva jediný parameter, handle portu získaný volaním funkcie `CreateFile`.

```
BOOL CloseHandle (
        HANDLE hObject
);
```

3 Multithreading v prostredí Windows

V multivláknovom prostredí Windows sa program môže rozdeliť na niekoľko samostatných častí, tzv. *threadov*, ktoré bežia súbežne. Ich využívanie vedie k efektívnejšiemu využívaniu procesoru a z pohľadu užívateľského rozhrania nám zaistí nezávislosť na vykonávanom výpočtovom výkone.

Vlákno je v programe reprezentované funkciou, ktorá môže volať ďalšie funkcie. Program začína pracovať hlavným vláknom, ktoré je vo Windows reprezentované funkciou *WinMain*. Ako náhle program beží, môžeme špeciálnym systémovým volaním vytvoriť ďalšie vlákna. Operačný systém potom preemptívne prepína riadenie medzi vláknami takmer rovnako, ako prepína medzi procesmi. Aplikácie s vláknami môžeme rozdeliť do troch modelov:

- **Jednoduché vlákna** (Single Threads) – Vlastnému procesu odpovedá iba jedno vlákno. Vlákno vykonáva všetku prácu procesu.
- **Oddelené vlákna** (Apartment Threads) – Kód oddeleného vlákna sa môže spustiť vo vlastnom vlákne, ale s obmedzením na oddelenú časť pamäte.
- **Voľné vlákna** (Free Threads) – Najkomplexnejší model práce s vláknami. Na rozdiel od oddelených vlákien, voľné vlákna nie sú obmedzené pamäťovým priestorom.

Implementácia použitia vlákien je rôzna, ako pre WinAPI, .NET alebo C++ Builder. Princíp vytvorenia vlákna však zostáva všade rovnaký. Vždy je treba definovať funkciu alebo metódu, na ktorej sa vlákno spustí. Vlákno je možné ukončiť, pozastaviť, spustiť. Každému vláknu je možné predávať dáta pomocou udalosti alebo pri inicializácii.

Pri použití vlákien nastáva veľký problém pri prístupe k zdieľaným objektom. Aplikácia musí zaistiť, aby nenastala chyba pri súčasnom čítaní a zápise hodnoty viacerými vláknami do rovnakého objektu. Na to je možné využiť monitor, kritickú sekciu alebo semafór.

Na vytvorenie vlákna sa vo Windows API používa funkcia `CreateThread`. Funkcia vytvorí vlákno vo virtuálnom adresovom priestore volajúceho procesu.

```
HANDLE WINAPI CreateThread (
      LPSECURITY_ATTRUBUTES lpThreadAttributes,
      SIZE_T dwStackSize,
      LPTHREAD_START_ROUTINE lpStartAddress,
      LPVOID lpParameter,
      DWORD dwCreationFlag,
      LPDWORD lpThreadId
);
```

Najdôležitejším parametreom je *lpStartAddress*, ktorý reprezentuje funkciu, na ktorej sa vlákno pustí. Predstavuje počiatočnú adresu vlákna. Funkcia má presne definovaný tvar, očakáva jeden parameter typu *LPVOID* (obecný ukazovateľ) a vracia hodnotu typu *DWORD*. Za zmienku stojí ešte parameter *dwCreationFlag*, ktorý udáva, či sa vlákno ihneď po vytvorení spustí. Hodnota *CREATE_SUSPENDED* spôsobí, že vlákno bude po vytvorení pozastavené, dokiaľ sa neuvoľní funkciou `ResumeThread`. Na opätovné zastavenie vlákna môžme použiť funkciu `SuspendThread`.

4 Meranie času v prostredí Windows

Vo Windows je možné zisťovať a merať časy rôznymi spôsobmi, ktoré sa líšia svojou presnosťou a použiteľnosťou. Táto kapitola uvádza ich stručný prehľad, výhody a nevýhody.

4.1 Časovače

Od verzie *Windows 2000* má programátor k dispozícii tri druhy časovačov. Štandardný (*Win32 Timer*), multimediálny, oneskorený (*Waitable Timer*) a časovač fronty (*Queue Timer*).

Štandardný časovač funguje na princípe zasielania správ aplikácii. Po vytvorení a nastavení periódy bude časovač pravidelne zasielať správy typu *WM_TIMER* po každej uplynutej perióde. Výhoda tohto časovača je v jednoduchosti používania, nevýhoda v pomerne vysokej nepresnosti. Hlavnou príčinou je spôsob, akým časovač dáva na vedomie uplynutie periódy. Správa *WM_TIMER* nemá voči ostatným správam vyššiu prioritu, preto musí čakať na spracovanie vo fronte správ. Alternatívne je možné pri vytváraní časovača špecifikovať funkciu, ktorá bude vykonaná po uplynutí periódy. Tento spôsob je ale rovnako založený na zasielaní správ. V najlepšom možnom prípade je presnosť štandardného časovača 55 ms.

Multimediálny časovač je v celku odlišný. Neposiela žiadne správy, namiesto toho volá predpísanú funkciu priamo (*callback funkcia*), preto je ďaleko presnejší. Prvý krok pri používaní multimediálneho časovača je nastavenie rozlíšenia. Rozlíšenie časovača určuje presnosť. Pri rozlíšení

treba dať pozor na to, že rôzne systémy podporujú rôzne minimálne hodnoty rozlíšenia. Ak je rozlíšenie príliš malé alebo volaná funkcia príliš rozsiahla, je zaťaženie procesoru pomerne vysoké, a preto je dobré používať multimediálny časovač opatrne.

Oneskorené časovače sú špeciálne objekty jadra operačného systému, ktoré môžu byť v stave „signalizovaný“ v určitý čas alebo v pravidelných časových intervaloch. Reakciou na zmenu stavu časovača je zavolanie funkcie, tzv. *completion routine*. Aby bolo možné túto funkciu zavolať, vlákno, v ktorom sa časovač používa, musí byť v čakajúcom stave (musí vykonávať jednu z funkcií `WaitForSingleObject`, `WaitForMultipleObjects`, `MsgWaitForMultipleObject` alebo `SignalObjectAndWait`). Z praktického hľadiska to znamená, že vlákno bude počas práce s časovačom zablokované.

Poslednými časovačmi použiteľnými vo Windows sú časovače front (*Queue Timer*). Sú to odľahčené objekty systémového jadra. Ako väčšina časovačov, umožňujú definovať callback funkciu, ktorá bude zavolaná v prípade dosiahnutia nastavenej periódy. Aj v tomto prípade musí byť vlákno v čakajúcom stave s tým rozdielom, že čakanie zabezpečuje priamo vlákno.

Spoločnou nevýhodou časovačov pri použití na meranie času je tzv. časová latencia, ktorá sa týka vlákien obecne. Callback funkcie sú totiž vo väčšine prípadov funkcie samostatného vlákna. Pri čakaní na nasledujúci časový interval je vlákno systémom uspané a po dobehnutí intervalu je nastavený príznak na jeho zobudenie. Problém spočíva v multitaskingu, ktorý toto vlákno neaktivuje okamžite, ale až po tom čo práve vykonávané vlákno spotrebuje jemu pridelený časový interval. Toto spôsobuje premenlivú nepresnosť pohybujúcu sa od 0 po niekoľko milisekúnd v závislosti na množstve vlákien v systéme.

4.2 Čítače

Moderné počítače obsahujú výkonový čítač (*performance counter*) s vysokým rozlíšením. Názov je odvodený od jeho pôvodného účelu, merania výkonnosti programov. Jeho prednosťou je rozlíšenie 1000 krát lepšie ako multimediálny časovač z predchádzajúcej kapitoly. Pracuje na frekvencii 1.1 až 1.2 miliónov tikov za sekundu. Presnú frekvenciu zistí funkcia `QueryPerformanceFrequency`. Obsah čítača zase funkcia `QueryPerformanceCounter`.

```
BOOL QueryPerformanceFrequency ( LARGE_INTEGER* lpFrequency );
BOOL OueryPerformanceCounter ( LARGE_INTEGER* lpPerformanceCount );
```

Nevýhodou ostáva, že tento čítač nemusí byť hardwarovo podporovaný na každom počítači a fakt, že frekvencia čítania je závislá na frekvencii CPU a tá sa môže pri notebookoch dynamicky meniť.

5 Stručný popis mikroprocesoru 2051

Mikroprocesor *ATMEL 89c2051* je kompatibilný s radou procesorov 8051. Ide o mikroprocesor s harwardskou architektúrou, pri ktorej je oddelená programová a dátová pamäť. Medzi jeho základné vlastnosti patrí:

- 2 KB Flash pamäť programu
- Napájanie v rozsahu 2.7 až 6 V
- Hodinový kmitočet v rozsahu 0 až 24 MHz
- 128 B dátovej RAM
- 15 vstup/výstupných liniek, skratový prúd jednej linky je 20 mA
- Dva 16 bitové čítače/časovače
- 6 zdrojov prerušenia
- Integrovaný sériový kanál
- Integrovaný analógový komparátor

Mikroprocesor obsahuje plne duplexný sériový kanál. Na prijímacej strane je jeden vysúvací register, ktorý umožňuje odoberať znaky s oneskorením zodpovedajúcim príjmu jedného znaku. Sériový kanál môže pracovať v štyroch rôznym módoch, ktoré je možné zvoliť nastavením bitov *SM1* a *SM0* v registri *SCON* a najvyššieho bitu v registri *PCON*.

Mód	***SM0***	***SM1***	***Režim***	***Prenosová rýchlosť***
0	0	0	8 bitový posuvný register	OSC/12
1	0	1	8 bitový asynchrónny prenos	čítač/časovač 1
2	1	0	9 bitový asynchrónny prenos	OSC/64 alebo OSC/32
3	1	1	9 bitový asynchrónny prenos	čítač/časovač 1

Tabuľka 5.1: Režimy sériového kanálu

Ako sériový kanál, tak aj čítače/časovače môžu pracovať v štyroch rôznych módoch, ktoré sa prepínajú nastavením bitov *M0* a *M1* v registri *TMOD*.

Mód	***M1***	***M0***	***Režim***
0	0	0	8 bitový čítač s 5 bitovou preddeličkou
1	0	1	16 bitový čítač
2	1	0	8 bitový čítač s prednastavením
3	1	1	8 bitový čítač + 8 bitový časovač

Tabuľka 5.2: Režimy čítačov/časovačov

6 Návrh

Druhá časť bakalárskej práce je venovaná konkrétnej implementácii digitálnych stopiek s ohľadom na požiadavky zadania a nasledujúcej podkapitoly.

6.1 Požiadavky

- **Presnosť** – Najdôležitejší parameter stopiek. Je daná presnosťou oscilátora, použitých elektronických súčiastok a technológie.
- **Rozsah** – Minimálny a maximálny možný čas, ktorý sú stopky schopné odmerať.
- **Použiteľnosť** – Možnosť praktického využitia v reálnom systéme.
- **Paralelizmus** – Stopky majú byť schopné paralelného merania dvoch časov bez možnosti ovplyvniť sa navzájom.
- **Užívateľské rozhranie** – Keďže ide o stopky pripojiteľné k počítaču, je prehľadné užívateľské rozhranie nutné.
- **Cena** - Je ďalším rozhodujúcim faktorom. Nemalo by sa ale zabúdať na to, že so znižovaním nákladov klesá aj celková kvalita zapojenia.

6.2 Možnosti

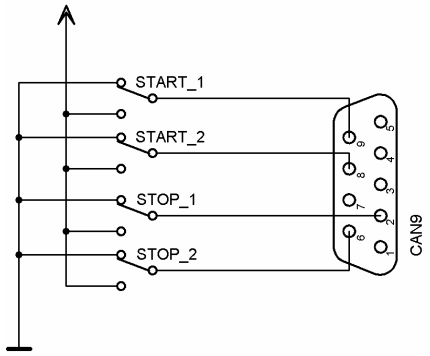
Pri návrhu som zvažoval viaceré varianty pripojenia kontaktov k počítaču, ich výhody a nevýhody.

Prvým z možných riešení je pripojenie kontaktov priamo na port počítača. Uvažovaný port by mal mať minimálne štyri nezávislé vstupy, dva pre štartovacie a dva pre stopovacie kontakty. Z tohto uváženia vyplýva, že môžeme použiť len sériový port. Paralelný port nie je vo Windows API podporovaný v obojsmernom režime, teda nemôže byť použitý ako vstupný. USB port neobsahuje potrebné množstvo vstupných pinov.

Sériový port má štyri programovo dostupné vstupy, *RxD, DSR, CTS, RI*. Možný spôsob zapojenia je na *obr. 6.1*. Výhodou tohto riešenia je veľmi jednoduché zapojenie kontaktov a nízka cena. Presnosť je závislá od aplikácie. Na aplikačnej úrovni je systém Windows schopný rozlíšenia len rádovo desiatok milisekúnd. Presné meranie času vo Windows je potrebné riešiť na úrovni jadra a systémových ovládačov.

Druhý, o niečo komplikovanejší návrh spočíva v pripojení špecializovaného hardwaru. Ten bude merať časové intervaly a namerané hodnoty zasielať do počítača na spracovanie. Hlavnou výhodou tohto spôsobu je pomerne vysoká presnosť.

Toto riešenie som vybral ako najvhodnejšie a rozhodol sa ho realizovať.

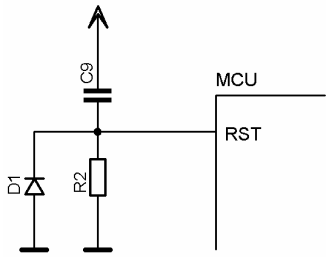


Obr. 6.1: Pripojenie kontaktov k sériovému portu

7 Hardwarová realizácia

7.1 Popis zapojenia

Jadro hardwarového modulu tvoria dva mikroprocesory *ATMEL 89c2051*. Ide o procesor kompatibilný s radou 51 s 2KB flash ROM a 128B RAM pamäte. Ako prevodník úrovní RS232 – TTL je použitý obvod *MAX232* od firmy *MAXIM* v katalógovom zapojení. Každý procesor je taktovaný vlastným kryštálom a má pripojený resetovací obvod.



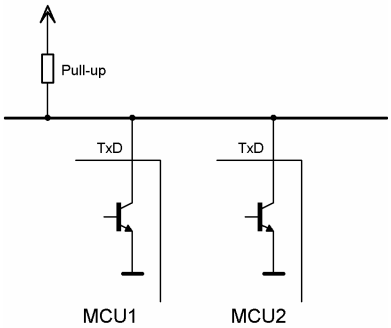
Obr. 7.1: Resetovací RC obovd

Po pripojení napájania sa na vstupe *RST* mikroprocesoru objaví napájacie napätie, ktoré s časovou konštantou $\tau = R2 \cdot C9$ klesá k nule. Procesor je resetovaný hodnotou logickej 1. Beh programu sa teda spustí pri napätí vstupu medzi minimálnou hodnotou log. 1 a maximálnou hodnotou log. 0, čo v prípade napájacieho napätia 5V zodpovedá intervalu 0.9 až 3.5V. Minimálnu dobu oneskorenia po spustení môžme spočítať podľa vzťahu *(7.2)*.

$$t = \tau \cdot ln \frac{5}{3.5} \qquad (7.2)$$

Pre hodnoty súčiastok $R2 = 100k\Omega$ a $C9 = 100nF$ je minimálna doba $t = 3.56ms$, čo postačuje k spoľahlivému resetu procesoru. Rýchle vybitie kondenzátoru po odpojení napájacieho napätia zaisťuje dióda D1.

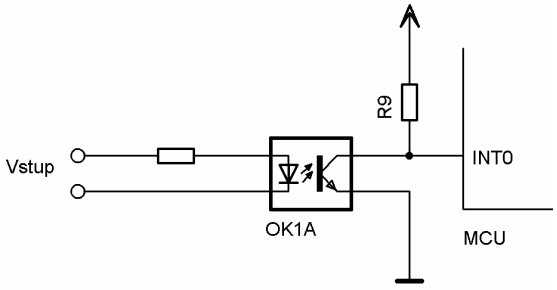
Spojenie výstupov *TxD* mikroprocesora do jednej zbernice bolo možné vďaka ich zapojeniu s otvoreným kolektorom. Vďaka tomu nemôže dôjsť k poškodeniu výstupných obvodov pri súčasnom vysielaní. Zdvíhacie (pull-up) rezistory udržiavajú na zbernici definovanú úroveň. V mojom prípade som pull-up odpory osadzovať nemusel, pretože sú integrované v obvode *MAX232*.



Obr. 7.3: Pripojenie na zbernicu

7.2 Pripojenie kontaktov

Pri návrhu som sa snažil o to, aby bolo možne pripojiť obecne čo najrôznejšie štartovacie/stopovacie kontakty. Vstupy sú preto galvanicky oddelené optočlenmi *PC847* (4x optočlen v puzdre DIL16, $V_{ISO} = 5000V$). Vstupné rezistory R10 – R14 je nutné voliť s ohľadom na predpokladané vstupné spínacie napätie. V zapojení je použitá hodnota $1k\Omega$, čo zodpovedá vstupnému napätiu 5 až 30V. Tieto hodnoty sú dané maximálnym prúdovým zaťažením optočlenu $I_F = 50mA$. Výstup je pripojený priamo na vstup vonkajšieho prerušenia procesoru (*INT0, INT1*).



Obr. 7.4: Pripojenie vstupných kontaktov

Pull-up rezistor R9 udržuje na vstupe procesoru úroveň log. 1, aby nedochádzalo k náhodnému výskytu prerušenia.

7.3 Popis funkcie

Po pripojení napájacieho napätia dôjde k inicializácií procesora a jeho periférií. Sériový kanál sa nastaví do 8 bitového režimu bez parity, rýchlosť je stanovená na 4800 Bd. Vynulujú sa registre časovača 0, povolí sa prerušenie od sériového kanálu, od čítača/časovača a oba vonkajšie prerušenia.

Privedením napätia na vstupné svorky sa otvorí fototranzistor optočlenu a na vstupe *INTx* sa objaví log. 0. Zmena úrovne signálu spôsobí vznik prerušenia. Podľa zdroja prerušenia sa časovač 0 spustí (*INT1*) alebo zastaví (*INT0*). Nameraná hodnota sa uloží a odošle po sériovom kanáli do počítača. Hodnoty, ktoré nie sú potvrdené ostávajú v pamäti a sú dostupné na vyžiadanie.

7.4 Konštrukcia DPS

Zariadenie je realizované na jednostrannej doske plošného spoja s rozmermi 90mm x 85mm. Sú použité súčiastky pre klasickú montáž. Táto technológia bola zvolená z dôvodu výrazne jednoduchšej montáže. Použitím technológie SMD by sa rozmery dosky zmenšili, ale praktický význam by to neprinieslo. Pre návrh bol použitý systém *Eagle 4.16 Light*.

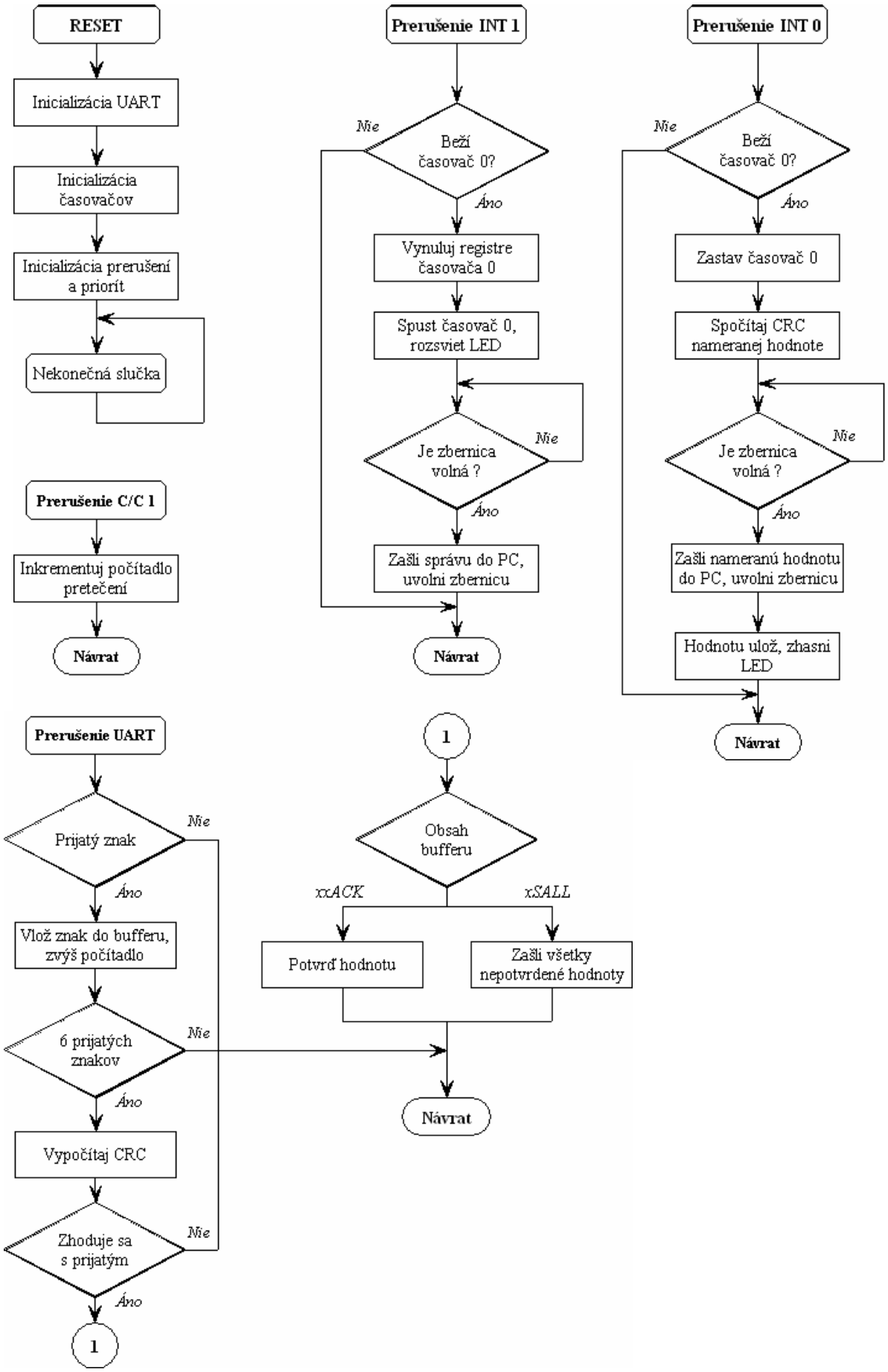
8 Softwarová realizácia

8.1 Firmware

Celý ovládací software je implementovaný v mikroprocesore. Ten zaisťuje presné meranie času, komunikáciu s počítačom a snímanie stavu vstupov. Program je pre oba procesory takmer totožný.

Ako programovací jazyk som volil jazyk C. Je to moderný spôsob programovania mikroprocesorov na rozdiel od asembleru, ktorý sa používa čoraz menej. Procesor *ATMEL 89c2051* jazyk C plne podporuje.

Program je tvorený jedným modulom *main.c*. Kvôli obmedzenej ROM pamäti nebolo možné použiť žiadne dostupné knižnice, pretože boli pamäťovo veľmi náročné. Vstup/výstupné rutiny sú preto obsiahnuté priamo v programe.

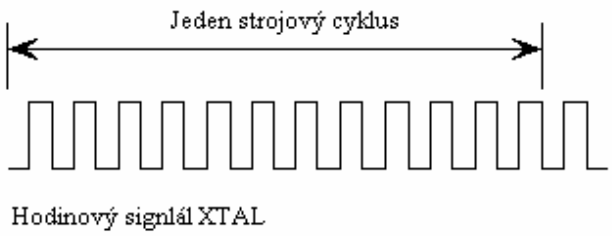


Obr. 8.1: Vývojový diagram riadiaceho softwaru

8.1.1 Meranie času

Neoddeliteľnou súčasťou mikroprocesorov sú čítače a časovače. Ich funkcia je rovnaká, ako pri iných zariadeniach číslicovej techniky. Čítač spravuje určité pamäťové miesto, ktorého obsah inkrementuje na základe zistenia nábežnej alebo zostupnej hrany sledovaného signálu. Funkcia časovača je rovnaká s tým rozdielom, že sledovaným signálom je v tomto prípade vnútorný signál so známou frekvenciou a periódou, tzv. hodinový signál.

Mikroprocesor *ATMEL 89c2051* obsahuje dva 16 bitové čítače/časovače (lomítko medzi čítačom a časovačom znamená, že môžu byť používané nezávisle v oboch funkciách). Pamäťovým miestom je tu register *TH0* a *TL0*, resp. *TH1* a *TL1* pre čítač/časovač 1. Jednotka sa k obsahu registrov *TH* a *TL* príslušného časovača pričíta za každý strojový cyklus, ktorý je tvorený 12 periódami oscilátoru.



Obr. 8.2: Strojový cyklus

Pri použití kryštálu X = 12MHz bude mať vstupný signál časovača frekvenciu 1MHz a periódu $1 \cdot 10^{-6}$ s. Inak povedané, obsah dvojice registrov *TH:TL* sa každú mikrosekundu inkrementuje. Ak je čítač/časovač v režime 1 (16 bitový časovač bez prednastavenia) a obsah registrov je nulový, tak presne o 2^{16} µs dôjde k pretečeniu časovača a nastaví sa príznakový bit *TF*. Ten spôsobí vznik žiadosti o prerušenie. V obslužnej rutine prerušenia nám neostáva nič iné, len pretečenia počítať.

```
void timer0_isr (void) interrupt 1
{
      timer_ticks = timer_ticks + 1;
}
```

Ak je časovač spustený, tak počas jednej minúty dôjde celkovo k $6 \cdot 10^7 \div 2^{16}$ prerušeniam, čo je približne 915 krát. Maximálny rozsah meraného času je preto daný použitým dátovým typom na ukladanie počtu prerušení. Ja som kvôli obmedzenej pamäti zvolil dvoj bajtový `unsigned int`. Horná hranica je v tomto prípade približne 71 minút. Spodná hranica je jeden tik časovača – 1µs. Jednoduchou zmenu dátového typu na `unsigned long` sa horná hranica výrazne posunie.

Celkový čas dostaneme, ak sčítame aktuálny obsah registrov *TH:TL* a počet pretečení vynásobený o hodnotu, pri ktorej k pretečeniu došlo. Pri 16 bitovom časovači je to už spomínaných 2^{16}.

```
useconds = timer_ticks * 65536 + ((unsigned int)(TH0 << 8 | TL0));
```

Presnosť celého zapojenia je daná presnosťou použitého kryštálu a odchýlkami súčiastok použitých v oscilačnom obvode mikroprocesoru. Použitý kryštál má maximálnu odchýlku ±20 ppm (Parts Per Milion) čo v najhoršom možnom prípade vedie k absolútnej odchýlke ±0,2 ms na jednu sekundu. Rozlíšenie stopiek je jeden strojový cyklus mikroprocesora, 1µs.

Spustenie a zastavenie časovača sa deje v obslužných rutinách vonkajších prerušení. *INT1* časovač spustí, *INT0* zastaví. Žiadosť o vonkajšie prerušenie je aktivovaná zostupnou hranou signálu na vstupe.

8.1.2 Ukladanie času

Pri každom ukončení merania sa nameraná hodnota zašle do počítača a uloží do pamäte mikroprocesoru. V prípade, ak by komunikácia s PC zlyhala a údaj by prišiel poškodený resp. neprišiel vôbec, neprídeme o neho.

Z dôvodu šetrenia pamäte a výpočtového výkonu sa namerané hodnoty neprepočítavajú na µs, ale sú ukladané v tvare: vyšší byte počítadla pretečení, nižší byte počítadla pretečení, *TH0* a *TL0*.

```
data_arr[index].tvalue =
(unsigned long)(((unsigned long)timer_ticks << 16) | (TH0 << 8) | TL0 );
```

Spolu s touto hodnotou sa ukladá aj príznak *ack*, ktorý informuje o tom, či bol odoslaný údaj potvrdený zo strany počítača a hodnota kontrolného súčtu *CRC*.

```
struct DTtable {
        unsigned long tvalue;
        uchar ack;
        uchar crc;
};
```

Maximálny počet časov, ktoré je možné takto uložiť je päť. Jednoduchou zmenou v programe je však možné tento počet rozšíriť podľa kapacity pamäte, ktorú máme k dispozícii. Hodnoty sa ukladajú v dátovej pamäti RAM procesoru, ktorá je napäťovo závislá. Pri prerušení napájania stráca svoj obsah a o namerané časy prichádzame. Tento problém by riešilo pripojenie externej EEPROM.

8.2 Komunikačný protokol

Keďže hardwarová časť neumožňuje žiadnym spôsobom zobraziť namerané hodnoty, sú údaje zasielané na sériový port počítača. Komunikácia sa riadi určitými pravidlami, ktoré sú uvedené v tejto kapitole.

Parametre prenosu:

- Rýchlosť: 4800 Bd (Najvyššia možná pri použití 12 MHz kryštálu)
- Parita: žiadna
- Počet dátových bitov: 8
- Počet stop bitov: 1

Celá komunikácia prebieha prostredníctvom správ. Mikroprocesor vysiela dva typy správ, obe s pevnou dĺžkou 7 bytov. Prvá správa sa zasiela pri zopnutí štartovacieho kontaktu a informuje užívateľské rozhranie o tom, že časovač je spustený. Druhá správa sa posiela pri zopnutí stopovacieho kontaktu a obsahuje práve odmeraný časový interval. Príklady oboch správ sú uvedené v tabuľkách *8.3a* a *8.3b*.

1	Číslo mikroprocesora (1 alebo 2)
'T'	"TIMON" (Timer On)
'I'	
'M'	
'O'	
'N'	
3	Kontrolný súčet CRC

Tabuľka 8.3a

1	Číslo mikroprocesora (1 alebo 2)
0	Poradové číslo času (0 až 4)
112	Vyšší byte počítadla pretečení
88	Nižší byte počítadla pretečení
220	TH0
134	TL0
25	Kontrolný súčet CRC

Tabuľka 8.3b

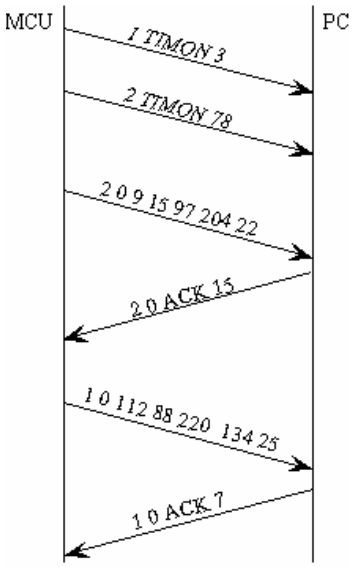
Na opačnej strane počítač správy prijíma a reaguje na ne. Na každú úspešne doručenú správu obsahujúcu časový interval reaguje odoslaním potvrdenia *ack*. Všetky hodnoty, ktoré neboli potvrdené zo strany počítača a nachádzajú sa v pamäti mikroprocesoru, sú dostupné na správu *sall*. Obe správy, ktoré zasiela počítač majú pevnú dĺžku 6 bajtov. Príklady sú zobrazené v tabuľkách *8.4a* a *8.4b*.

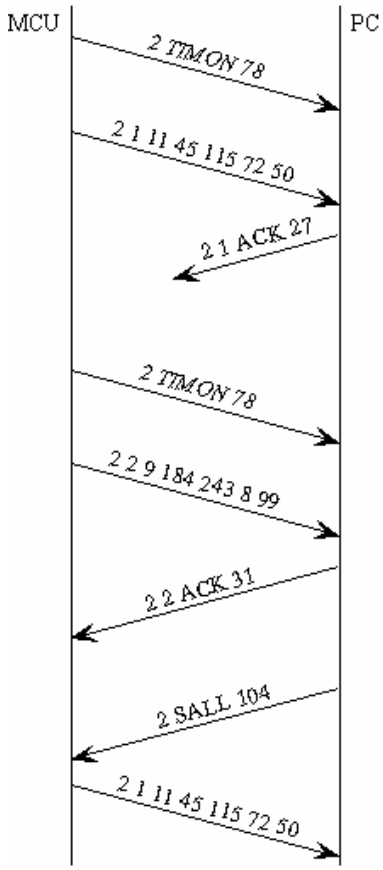
1	Číslo mikroprocesora (1 alebo 2)
2	Poradové č. potvrdzovaného času
´A´	"ACK" (Acknowledge)
´C´	
´K´	
12	Kontrolný súčet CRC

Tabuľka 8.4a

2	Číslo mikroprocesora (1 alebo 2)
´S´	"SALL" (Send All)
´A´	
´L´	
´L´	
104	Kontrolný súčet CRC

Tabuľka 8.4b





Obr. 8.5: Možné priebehy komunikácie

8.2.1 Zabezpečenie komunikácie

Z dôvodu vyššej odolnosti voči chybám vzniknutým pri prenose je na oboch stranách implementovaný zabezpečovací algoritmus *CRC* (Cyclic redundancy check). Ide o špeciálnu hašovaciu funkciu, ktorej výsledok sa zašle spolu s prenášanými dátami. Výsledkom funkcie je zvyšok po dvojkovom delení bez prenosu (carry). Po prijatí dát je kontrolný súčet opäť vypočítaný. Ak vyjde rôzny CRC správa sa považuje za chybnú a ignoruje sa.

Na zabezpečenie je použitý 8 bitový CRC algoritmus s generujúcim polynómom $C(x) = x^8 + x^2 + x + 1$. S ohľadom na rýchlosť algoritmu som zvolil jeho variantu s preddefinovanou tabuľkou. V mikroprocesore je tabuľka uložená v pamäti programu, pretože do dátovej časti by sa nevošla.

8.2.2 Prístup na zbernicu

Medzi oboma procesormi musia platiť určité pravidlá pri prístupe na zbernicu. Nemôže sa stať aby vysielali oba súčasne, pretože by došlo ku kolízii signálov. Z toho dôvodu je v zapojení implementovaný jednoduchý systém, ktorý riadi vysielanie na zbernicu.

Oba mikroprocesory sú prepojené jedným riadiacim signálom *CBUS*. Úroveň signálu log. 1 znamená, že zbernica je voľná a mikroprocesor môže vysielať. Naopak, úroveň log. 0 signalizuje obsadenie zbernice. Pred každým vysielaním procesor skontroluje stav riadiaceho signálu, obsadí zbernicu a po zaslaní správy zbernicu uvoľní.

Príjem správ nie je nijako obmedzený a oba procesory môžu čítať dáta zo zbernice súčasne.

```
while (!CBUS) ;          /* Check if the bus is available */
CBUS = 0;                /* Cast the bus */
...                      /* Transmitt the message */
CBUS = 1;                /* Free the bus */
```

8.3 Užívateľské rozhranie

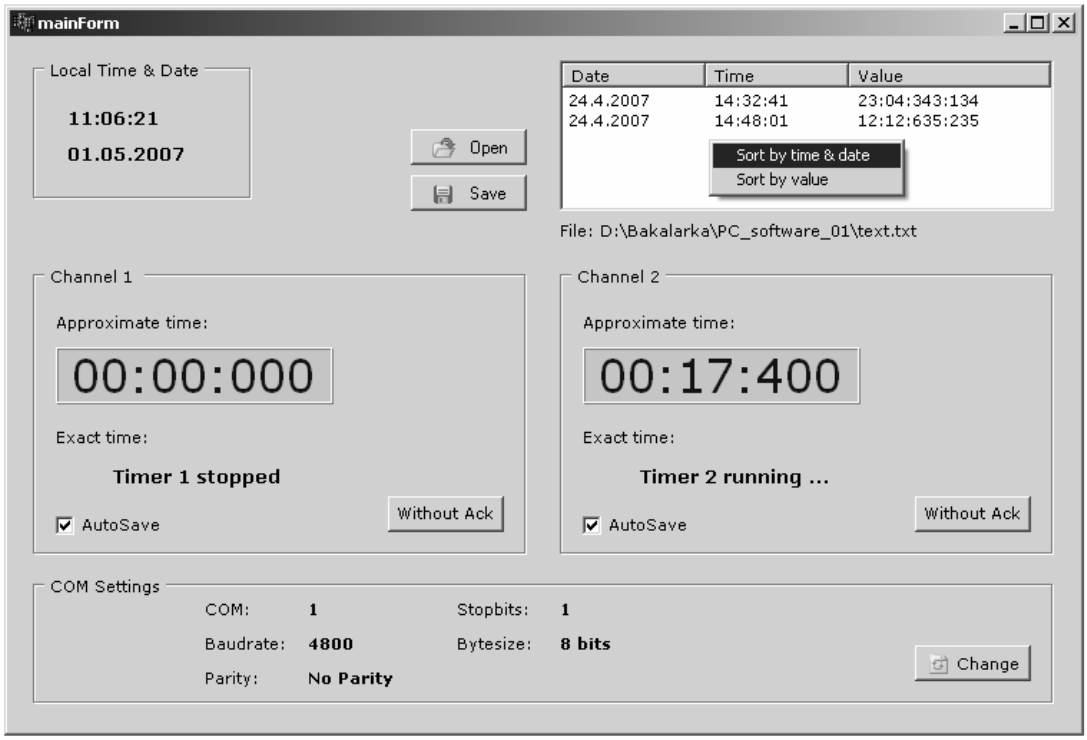
Úlohou užívateľského rozhrania je prijímať správy od hardwarového modulu, spracovať a zobraziť ich. Prijaté údaje je možné prezerať alebo uložiť do textového súboru. Počas merania je neustále zobrazovaná priebežná hodnota meraného času.

Rozhranie je napísané vo vývojovom prostredí *C++ Builder* od firmy *Borland*. Okno aplikácie je zobrazené na *Obr. 8.6*. Pre spustenie programu je treba, aby bol počítač vybavený aspoň jedným sériovým portom. Program sa pokúsi otvoriť port špecifikovaný v inicializačnom súbore (*.INI*). Užívateľské rozhranie nijako nekontroluje, či je hardwarový modul pripojený k počítaču, pretože ten ho k svojej činnosti výslovne nevyžaduje.

Program pozostáva s dvoch modulov: *serial.cpp*, ktorý zabezpečuje sériovú komunikáciu a hlavný modul *timer_source.cpp*. Výsledkom prekladu a linkovania je jeden spustiteľný *.exe* súbor. Ten obsahuje aj behové balíky (*run-time packages*) používané aplikáciami vytvorenými

v *C++Builderi,* preto je možné aplikáciu spustiť aj na počítači, na ktorom sa toto vývojové prostredie nenachádza. Za univerzálnu funkčnosť však zaplatíme o niečo väčšou veľkosťou *.exe* súboru.

Jednou z požiadaviek bolo, aby aplikácia pozostávala z viacerých vlákien. Vo výsledku je funkčnosť aplikácia rozdelená do dvoch vlákien. Jedno obstaráva komunikáciu s pripojeným modulom, prijíma a zasiela správy, druhé riadi chod celého grafického rozhrania.



Obr. 8.6: Okno aplikácie

- **Local Time & Date** – Lokálny čas a dátum. Ukladá sa spolu s nameranými hodnotami.
- **Open** – Otvorí súbor s časovými údajmi.
- **Save** – Uloží namerané hodnoty do textového súboru.
- **Approximate time** – Priebežný čas počas merania s presnosťou na stovky milisekúnd.
- **Exact time** – Presná nameraná hodnota.
- **AutoSave** – Pokiaľ je zaškrtnuté, nameraná hodnota je automaticky uložená do aktuálne otvoreného súboru. V opačnom prípade je hodnota zobrazená v zozname, ale nie je ukladaná. Treba tak urobiť manuálne tlačidlom *Save*.
- **Without Ack** – Vynúti si zaslanie všetkých doposiaľ nepotvrdených hodnôt z pamäte procesoru.
- **Change** – Zmena parametrov. Praktický význam má iba zmena portu (*COM1*, *COM2*, ...). Zmena iných parametrov má za následok nefunkčnú komunikáciu s modulom.

8.3.1 Implementácia

Aplikácia sa po spustení pokúsi otvoriť konfiguračný súbor *timer_project.ini*. Význam položiek je nasledujúci:

```
[PORT]                    ; sekcia nastavenia parametrov komunikácie
Port=1                    ; číslo sériového portu (1=COM1, 2=COM2)
Overlapped=false          ; overlapped režim (true/false)
[PATH]
File=D:\Timer\tims.txt    ; súbor, ktorý sa načíta pri štarte
```

Ak nie je konfiguračný súbor k dispozícii, použijú sa implicitné nastavenia: port *COM1*, overlapped režim a súbor *tims.txt* v aktuálnom adresári.

Po úspešnom otvorení sériového portu sa vytvorí nové vlákno, ktorého úlohou je komunikovať s hardwarovým modulom. Vlákno čaká na príjem znaku funkciou `WaitCommEvent`. V prípade, že je port otvorený v nonoverlapped režime, spôsobí táto funkcia zablokovanie vlákna, dokiaľ nebude prijatý aspoň jeden znak. Pokiaľ je vlákno zablokované, nie je možné vysielať na sériový port, a teda nie je možné dotazovať sa na časy uložené v pamäti mikroprocesoru. Ak sa aj napriek tomu pokúsime zaslať správu, bude doručená až po odblokovaní vlákna.

Ak je port otvorený v overlapped režime, funkcia `WaitCommEvent` okamžite vráti návratovú hodnotu a nespôsobí zablokovanie vlákna. Prijatie znaku je potom signalizované zmenou stavu udalosti *hEvent* štruktúry *OVERLAPPED*, ktorá je tretím parametrom funkcie `WaitCommEvent`.

```
hArray[0] = CreateEvent(NULL, FALSE, FALSE, NULL);
WaitCommEvent(hndl, &EvtMask, &ovrlpd);
hArray[1] = ovrlpd.hEvent;
dwRes =  WaitForMultipleObjects(2, hArray, FALSE, INFINITE);

switch (dwRes) {
       case WAIT_OBJECT_0 + 0:
       Transmit();
       break;

       case WAIT_OBJECT_0 + 1:
       Receive();
       break
}
```

Funkciou `CreateEvent` najskôr vytvorím novú udalosť, ktorá reprezentuje požiadavku o zápis na port. Funkcia `WaitForMultipleObjects` spôsobí zablokovanie vlákna a čaká, pokiaľ jedna z udalostí nezmení svoj stav. Podľa návratovej hodnoty určíme, ktorá z udalostí spôsobila odblokovanie vlákna.

Vlákno musí o prijatí správy informovať grafické rozhranie. Keďže v C++ Builderi nie je možné pristupovať z vlákien vytvorených funkciou `CreateThread` ku komponentom hlavného vlákna, je komunikácia riešená prostredníctvom zasielania správ.

```
SendMessage(mainForm->Handle, WM_TIMER_SET, TIM1, START);
```

Nameraný časový údaj je hlavnému vláknu prístupný v zdieľanej premennej `rxtime`. Aby sa predišlo problémom pri súčasnom zápise a čítaní, je prístup k premennej synchronizovaný pomocou kritickej sekcie (*critical section*).

Priebežný čas, ktorý aplikácia zobrazuje počas merania, je zaokrúhlený na stovky milisekúnd. Jeho meranie zabezpečuje samotné grafické rozhranie pomocou štandardného časovača Windows (*Win32 Timer*). Presná nameraná hodnota je zobrazená a ukladaná v tvare:

minúty:sekundy:milisekundy:mikrosekundy, napr. *22:12:343:672*

Spolu s nameranou hodnotou sa do textového súboru ukladá aj čas a dátum, kedy bolo meranie uskutočnené.

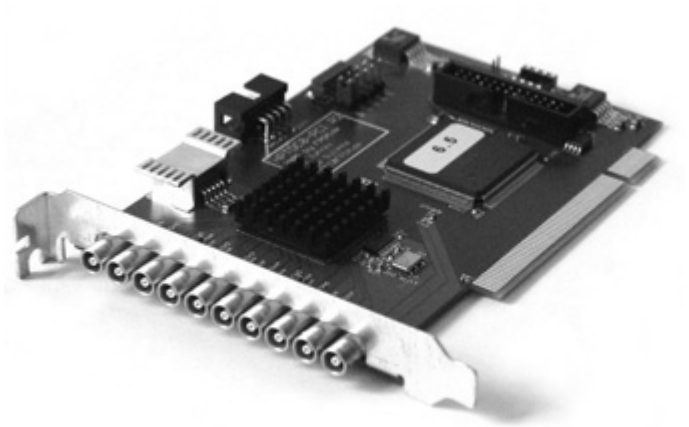
9 Porovnanie

Ďalším možným spôsobom ako vo Windows presne merať čas, sú *RTX* ovládače (Real Time Extension). Ide o systém založený na ovládačoch, ktorý umožňuje dostať Windows do pozície realtimového operačného systému. Realtimové systémy pracujú v reálnom čase a majú uplatnenie všade tam, kde je potreba okamžitej odozvy na podnety. RTX ovládače dokážu skrátiť dobu reakcie na 3µs.

RTX ovládače od firmy *Ardence* sa vyznačujú plnou podporov Windows API. Skrátenými reakciami systému je takto možné presné meranie času aj pomocou časovačov Windows.

Nevýhodou takéhoto riešenia je obmedzený počet realtimových úloh, ktoré je možné v systéme spustiť a cena. Pri zrovnateľnej presnosti je niekoľkonásobne vyššia.

Z hardwarových riešení by som chcel spomenúť zásuvné karty *TDC8* a *HPTDC8* od firmy *Cronologic*. Ide o zásuvné karty do PCI slotu určené predovšetkým na presné meranie času. Obe verzie sa o seba odlišujú presnosťou. Kým pri *TDC8* je presnosť 15ns, *HPTDC8* udáva presnosť 5ns. Karty sa dodávajú bez potrebného užívateľského rozhrania, čo ich znevýhodňuje oproti realizovaným stopkám.



Obr. 9.1: HPTDC8

10 Záver

Cieľom bakalárskej práce bolo navrhnúť a realizovať aplikáciu, ktorá umožní meranie dvoch časov pomocou externého štartovacieho a zastavovacieho impulzu. Hlavnou ideou bolo, aby meranie nevykonávala samotná aplikácia, ale externe pripojený hardware. Táto myšlienka bola pri návrhu ďalej rozvíjaná. Výsledkom je systém zložený z dvoch častí. Hardwarového modulu, ktorý zabezpečuje meranie času a aplikačného rozhrania, ktoré namerané hodnoty zobrazuje a spracováva.

Základom hardwarového modulu sú dva mikroprocesory *ATMEL 89c2051*. Každý obsahuje dva čítač/časovače, no prakticky použiteľný je iba jeden z nich, pretože jeden sa použije na generovanie rýchlosti pre sériový kanál. Použitím dvoch mikroprocesorov som navyše dosiahol maximálny paralelizmus, čo bolo požadované zadaním. Najdôležitejším parametrom stopiek je presnosť. V najhoršom možnom prípade sa mi podarilo dosiahnuť presnosť ±0.2ms na jednu sekundu. Táto hodnota je však iba teoretická a závisí na presnosti použitého kryštálu. Prakticky sa mi presnosť nepodarilo odmerať, pretože som nemal k dispozícii dostatočne presné referenčné stopky. Rozsah je definovaný ako minimálna a maximálna hodnota, ktorú je možné odmerať. Maximálna hodnota je obmedzená použitým dátovým typom na 71 minút. Jednoduchou zmenou programu je však možné túto hranicu výrazne zvýšiť. Pre moje potreby mi súčasná hodnota plne postačuje. Minimálna hodnota je opäť iba teoretická: 1µs. Mechanickými kontaktmi totiž nie je možné takéto krátke časové intervaly realizovať.

Prehľadné užívateľské rozhranie bolo nutné, pretože hardwarový modul neumožňuje zobraziť namerané hodnoty. Jedinou požiadavkou na aplikáciu bolo, aby pozostávala z viacerých vlákien. Funkčnosť je teda rozdelená do dvoch samostatných vlákien. Jedno zabezpečuje komunikáciu s pripojeným modulom a druhé chod celej aplikácie. V takomto prípade nemôže dôjsť k „zamrznutiu“ aplikácie v dôsledku chýb pri komunikácii a grafické rozhranie tak neustále reaguje na podnety užívateľa.

Najväčšou prednosťou celého zapojenia je fakt, že hardwarový modul nemusí byť počas merania vôbec pripojený k počítaču. Mikroprocesor si dokáže uschovať v pamäti päť naposledy nameraných časov a na požiadanie ich zaslať do počítača.

V dobe vysokej konkurencie je cena jedným z rozhodujúcich faktorov. Výrobné náklady hardwarového vybavenia neprekročili 1000 Kč, čo je v porovnaní s podobnými zariadeniami výborný výsledok.

Do budúcna plánujem rozšíriť aplikáciu o možnosť práce s XML formátom dát a hardwarovú časť o USB rozhranie. Väčšina dnes vyrábaných notebookov už sériový port nemá. Alternatívou je použitie USB – RS232 prevodníku.

Literatúra

[1] Kainka, B. *Využití rozhraní PC*. HEL Ostrava, 1996.

[2] Matoušek, D. *Udělejte si z PC* 2.díl. BEN – Technická literatúra Praha, 2002.

[3] Denver, A. *Serial Communications in Win32* [online]. [rev. 1995-11-12], [cit. 2007-05-07]. Dostupné z URL: <*http://msdn2.microsoft.com/en-us/library/ms810467.aspx*>

[4] ATMEL. *AT89C2051 Datasheet* [online]. [cit. 2007-05-07]. Dostupné z URL: <*http://www.atmel.com/dyn/resources/prod_documents/doc0368.pdf*>

[5] Hrbáček, J. *Komunikace mikrokontroléru s okolím* 1.díl. BEN – Technická literatúra Praha, 1999.

[6] Burkhard, M. *C pro mikrokontroléry*. BEN – Technická literatúra Praha, 2003.

[7] KEIL. *Cx51 User's Guide* [online]. [rev. 2004-01-01], [cit. 2007-05-07]. Dostupné z URL: <*http://www.keil.com/support/man/docs/c51*>

[8] MSDN library. *Multithreading with C and Win32* [online]. [cit. 2007-05-07]. Dostupné z URL: <*http://msdn2.microsoft.com/en-us/library/y6h8hye8(VS.71).aspx*>

[9] Kadlec, V. *Učíme se programovat v Borland C++Builder a jazyce C++*. Computer Press Brno, 2004.

[10] GES Electronics. *Katalog součástek 2006*. GES Plzeň, 2006.

Zoznam príloh

Príloha 1. Schéma zapojenia

Príloha 2. Obrazec plošného spoja, Osadzovací obrazec DPS

Príloha 3. Zoznam súčiastok

Príloha 4. Fotografie hotového modulu

Príloha 5. CD

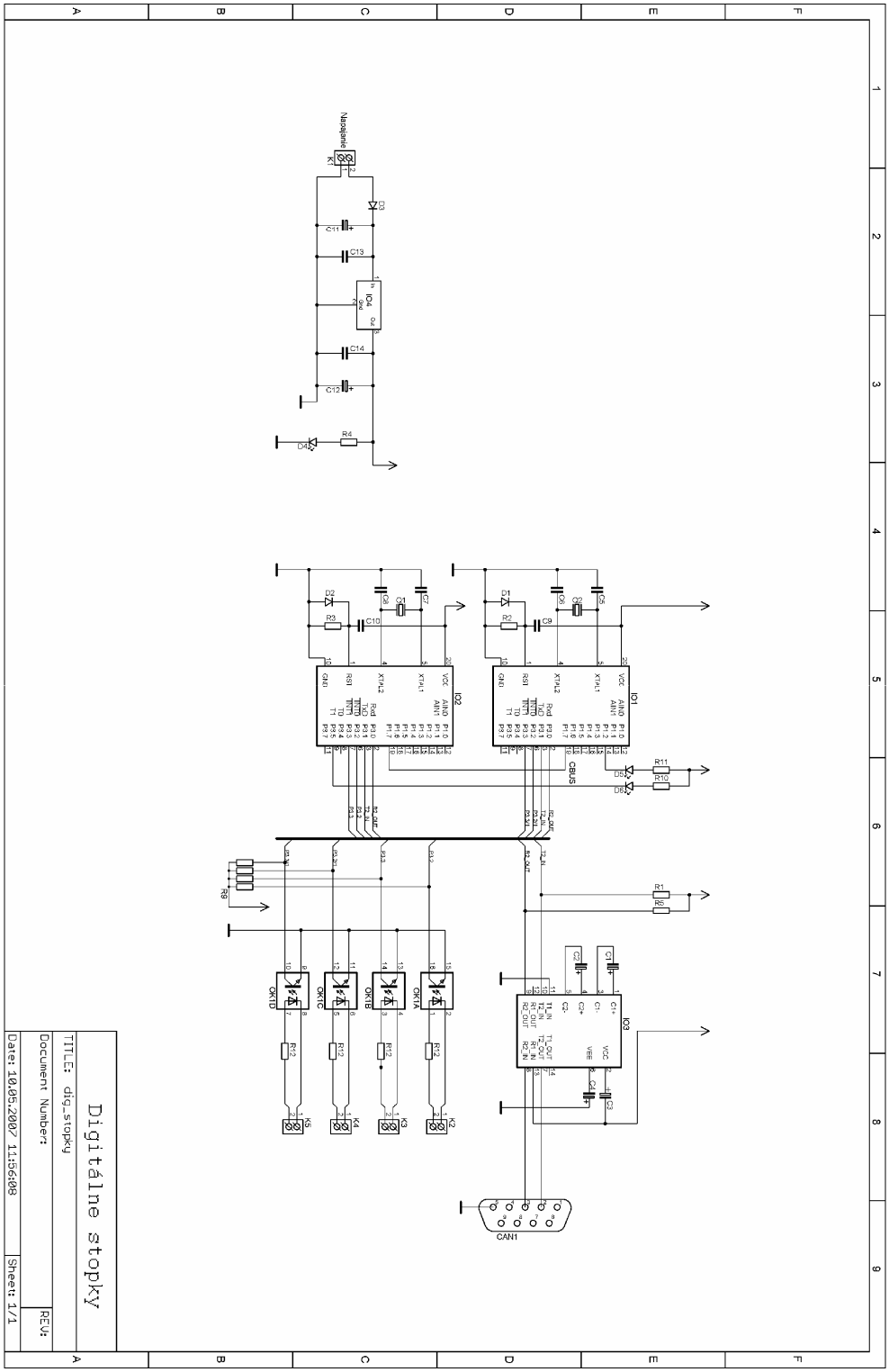
Obsah CD

1. Elektronická verzia bakalárskej práce
2. Zdrojové kódy užívateľského rozhrania v programe *C++ Builder 6.0*
3. Zdrojové kódy riadiaceho firmware pre mikrokontrolér *ATMEL 89c2051*
4. Schéma zapojenia a obrazec plošného spoja v programe *Eagle 4.16*.
5. Schéma zapojenia a obrazec plošného spoja vo formáte PNG.

Prílohy

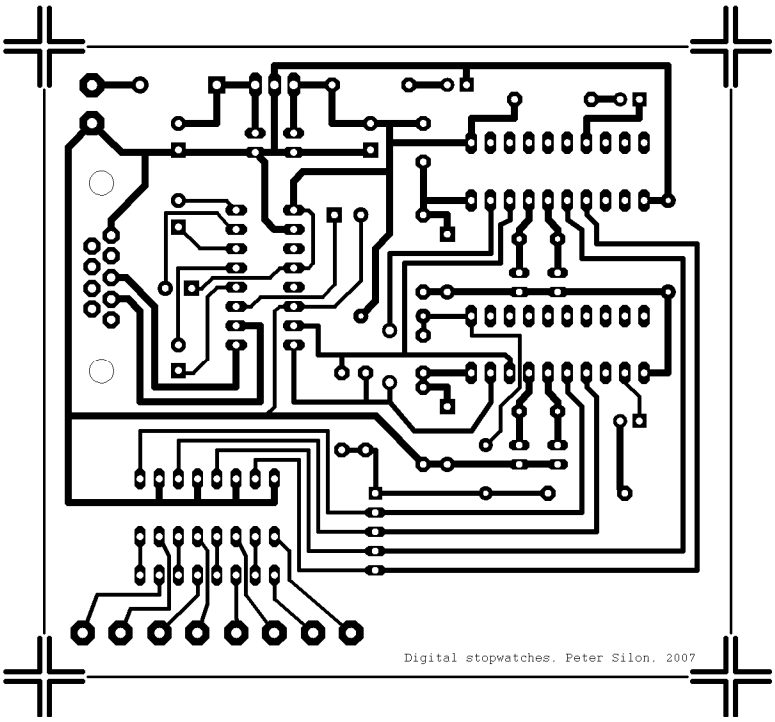
Príloha 1

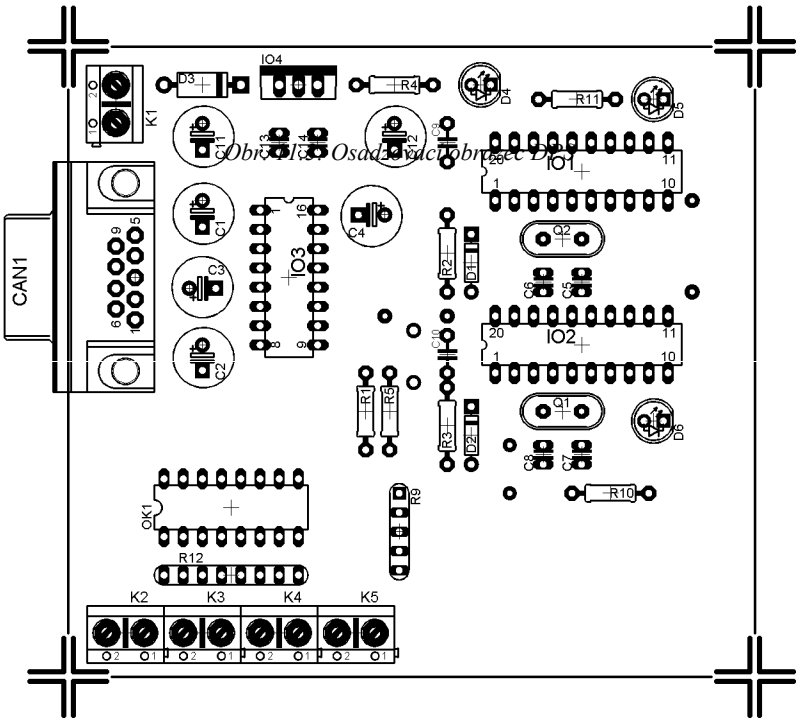
Schéma zapojenia



Príloha 2

Obrazec plošného spoja a osadzovací obrazec DPS





Obrazec a Osadzovací obrazec DPS

Príloha 3

Zoznam súčiastok

Názov	Hodnota	Poznámka
C1, C2, C3, C4	22 µF/16V	
C5, C5, C7, C8	33pF	
C9, C10, C13, C14	100nF	
C11	100uF/25V	
C12	47 µF/25V	
CAN1	CANNON 9	Uhlový 90° do DPS
D1, D2	1N4148	
D3	1N4007	
D4, D5, D6	LED	5mm
IO1, IO2	AT89C2051	Puzdro DIL 20
IO3	MAX232	Puzdro DIL 16
IO4	LM7805	
K1, K2, K3, K4, K5	Svorkovnica ARK	
OK1	PC847 (TLV847)	
Q1, Q2	Kryštál 12MHz	Puzdro HC49-U
R1, R5	10k	
R2, R3	100k	
R4, R10, R11	330R	
R9	4 x 10k	Odporová sieť so spol. vývodom
R12	4 x 1k	Odporová sieť bez spol. vývodu

Príloha 4

Fotografie hotového modulu