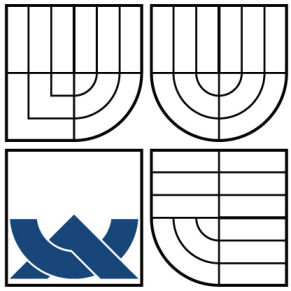
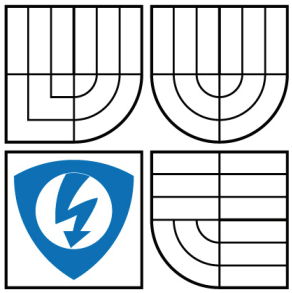


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE APROXIMAČNÍ FUNKCE DO MIKROPROCESORU PIC

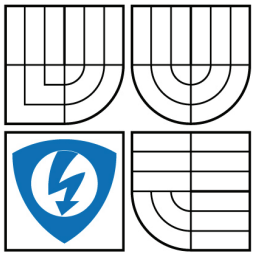
APPROXIMATION FUNCTION IMPLEMENTATION INTO PIC MICROPROCESSOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ PETŘÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ ŠMIRG
SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Tomáš Petřík ***ID:*** 70059
Ročník: 3 ***Akademický rok:*** 2008/2009

NÁZEV TÉMATU:

Implementace aproximační funkce do mikroprocesoru PIC

POKYNY PRO VYPRACOVÁNÍ:

Cíl práce se bude zabývat implementací některé aproximační funkce (např. metody nejmenších čtverců) v procesorech PIC. Účelem je pochopit problematiku mikrokontrolerů, jejich architekturu a využití v oblastech průmyslu. Výsledkem bakalářské práce bude implementace dané aproximační funkce do procesoru a její otestování.

DOPORUČENÁ LITERATURA:

[1] HRBÁČEK J.: Moderní učebnice programování mikrokontrolérů PIC - 1 díl, BEN, 2004, 96s. Technická literatura, ISBN 8073001365
[2] VACEK V.:Učebnice programováni PIC, BEN, 144s. Technická literatura, ISBN 80-86056-87-2

Termín zadání: 9.2.2009 ***Termín odevzdání:*** 2.6.2009

Vedoucí práce: Ing. Ondřej Šmirg

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práve třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

ANOTACE

Tato bakalářská práce se zabývá implementací aproximační funkce do mikroprocesoru PIC. Zaměřuje se na architekturu mikroprocesorů a rozbor aproximačních funkcí. Pro samotnou implementaci je pak vybrána metoda nejmenších čtverců a to z důvodu její relativní jednoduchosti a přesnosti v porovnání s ostatními metodami. Mikroprocesor má pomocí A/D převodníku změřit závislost nelineárního systému, jenž je k mikroprocesoru připojen. Měří se napětí na logaritmickém potenciometru, který reprezentuje nelineární systém, v závislosti na poloze jezdce odporové dráhy. Poté, pomocí zvolené metody, mikroprocesor dopočítá charakteristiku nelineárního systému. Naměřené hodnoty se porovnávají s teoretickými, které byly dopočítaný zvolenou metodou. Hodnoty naměřené, teoretické a jejich diference se pak zobrazují na displeji, připojeném rovněž k mikroprocesoru. V závěru se pak diskutují výsledky spolu s možností dalšího rozvinutí této práce.

KLÍČOVÁ SLOVA

PIC24HJ32GP202, aproximační funkce, metoda nejmenších čtverců, lineární regrese, logaritmická funkce.

ABSTRACT

This bachelor‘s thesis deals with the implementation of approximation function into the PIC microprocessor. Thesis is focused on microprocessor architecture and analysis of approximation functions. Then for actual implementation is chosen the method of least squares, and because of its relative simplicity and accuracy in comparison with other methods. Microcontroller is using the A/D converter to measure the dependence of non-linear system, which is connected to the microprocessor. Measure the voltage on a logarithmic potentiometer, which represents non-linear system, depending on the position of the slider track resistance. Then, using the method chosen microprocessor calculates the characteristic non-linear system. The measured values are compared with the theoretical, calculated the chosen method. The values measured, and their theoretical difference is then displayed on the display, connected to the microprocessor. In conclusion, then discuss the results together with the possibility of further developing this work.

KEYWORDS

PIC24HJ32GP202, approximation function, the method of least squares, linear regression, logarithmic functions.

PETŘÍK, T. *Implementace aproximační funkce do mikroprocesoru PIC*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 46 s. Vedoucí bakalářské práce Ing. Ondřej Šmirg.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Implementace aproximační funkce do mikroprocesoru PIC“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení §152 trestního zákona č. 140/1961 Sb.

V Brně dne …… ……..

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu mé bakalářské práce Ing. Ondřeji Šmirgovi, za velmi užitečnou metodickou pomoc a cenné rady při zpracování bakalářské práce.

V Brně dne …… ……..

(podpis autora)

Obsah

Úvod ... 10
1 Mikrokontroléry PIC ... 11
1.1 Základní popis ... 11
1.2 Organizace paměti ... 13
1.2.1 Organizace programové paměti ... 13
1.2.2 Registry PCL a PCLATH ... 14
1.3 Vstupně/výstupní brány ... 14
1.4 Čítače/Časovače ... 15
1.4.1 TIMER0 ... 15
1.4.2 TIMER1 ... 15
1.4.3 TIMER2 ... 16
1.5 Analogově-digitální převodník ... 16
2 Aproximace funkce ... 17
2.1 Rozdělení aproximačních funkcí ... 17
2.2 Metoda nejmenších čtverců ... 19
2.2.1 Lineární regrese (aproximace přímkou) ... 19
2.2.2 Polynomická regrese (aproximace polynomem) ... 22
2.2.3 Kvadratická regrese (aproximace parabolou) ... 24
2.2.4 Srovnání metod nejmenších čtverců ... 26
3 Schéma zapojení a postup měření ... 27
4 Vývoj a ladění zdrojového kódu ... 29
4.1 Základní nastavení mikroprocesoru ... 29
4.2 Velikost a optimalizace kódu ... 30
4.3 Popis funkce pro výpočet aproximační funkce ... 31
5 Návrh DPS pro obvod s mikroprocesorem ... 33
6 Naměřené a vypočtené hodnoty ... 34
Závěr ... 37
Seznam použité literatury ... 39
Příloha A ... 41
Příloha B ... 46

Seznam obrázků

Obr. 1.1Diagram mikrokontroléru PIC24HJ32GP202 11
Obr. 1.2 Harvardská koncepce 11
Obr. 1.3 Von Neumannova koncepce 12
Obr. 1.4 Blokový diagram mikrokontroléru PIC24HJ32GP202 [1] 12
Obr. 2.1 Příklad interpolace lineárním splajnem 18
Obr. 2.2 Příklad lineární regrese 18
Obr. 2.3 Odchylky e_i 19
Obr. 2.4 Hledání rovnice, pro niž je součet obsahů čtverců nejmenší 20
Obr. 2.5 Aproximace polynomem 23
Obr. 2.6 Aproximace parabolou 25
Obr. 7 Schéma zapojení pro měření 27
Obr. 8 Převodní charakteristika IC/IF = f (IF) při TA = 25°C, VCE = 5.0 V 28
Obr. 9 Prostředí programu MPLAB v8.30 29
Obr. 10 Základní nastavení mikroprocesoru 30
Obr. 11 Využití paměti programu a paměti dat mikroprocesoru 30
Obr. 12 Návrh zapojení v programu Formica Schematic 33
Obr. 13 Návrh DPS v programu Formica Layout 33
Obr. 14 Závislost napětí na poloze jezdce U = f(α) pro prvních 5 uzlových bodů 35
Obr. 15 Závislost napětí na poloze jezdce U = f(α) pro všechny uzlové body 35
Obr. 16 Srovnání naměřených a teoretických hodnot napětí 36

Seznam tabulek

Tab. 1 Naměřené hodnoty voltmetrem 34
Tab. 2 Naměřené hodnoty mikroprocesorem PIC24HJ32GP202 34
Tab. 3 Naměřené hodnoty mikroprocesorem PIC - kompletní měření 36
Tab. 4 Hodnoty vypočítané dle aprox. fce získané z mikroprocesoru PIC 36

Úvod

Zadaná práce je zaměřena na využití mikroprocesoru PIC v průmyslu, kde se pomocí tohoto procesoru budou měřit veličiny a na základě naměřených hodnot těchto veličin se budou počítat jejich funkce. Úkol by byl též realizovatelný na klasickém PC se speciálním hardwarovým a softwarovým vybavením, avšak mnohem výhodnější variantou je užití jednočipu. Cena potřebného vybavení pro realizaci tohoto zadání je v případě užití jednočipu výrazně nižší než u varianty s PC. Právě proto jsou mikroprocesory vhodné pro spoustu různých měřicích aplikací, sběr menšího objemu dat a řízení a ovládání jednodušších úloh, jednodušších výrobních procesů, atd. Navíc na mnoha místech, kde je potřeba nějaké jednodušší realizace řízení, ovládání či měření, bývají obvykle horší pracovní podmínky způsobené především vyšší prašností prostředí, vyšší teplotou apod. V takovémto případě není vhodné užití klasického PC a bylo by potřeba sáhnout po dražší variantě speciálního průmyslového PC nebo v průmyslu hojně využívaných PLC (př. Siemens, Mitsubishi). V případě jednodušších aplikací je tedy z ekonomického hlediska lepší využít mikrokontrolerů.

K řešení této problematiky je nutná znalost architektury mikroprocesoru PIC a orientace v aproximačních funkcích.

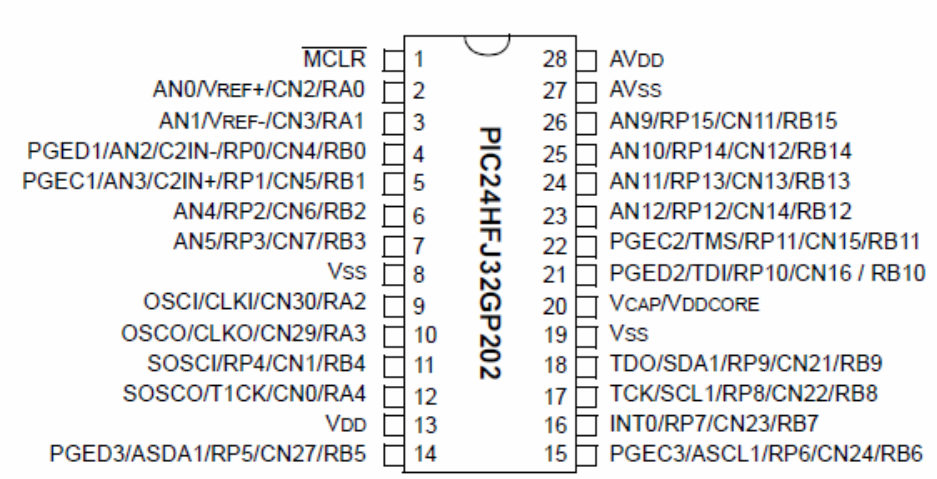
Úkolem je navržení vhodného zapojení mikroprocesoru spolu s ostatními periferiemi pro proměření charakteristiky nelineárního potenciometru. Po změření počátečních několika uzlových bodů charakteristiky daného potenciometru by měl být dále mikroprocesor schopen vyjádřit teoretickou funkci daného potenciometru a na základě této funkce dopočítat předpokládaný následný průběh závislosti potenciometru. Pro výpočet funkce potenciometru je potřeba provést rozbor aproximačních funkcí a na základě tohoto rozboru zvolit nejvhodnější metodu pro řešení tohoto problému. Je potřeba zohlednit složitost implementace do mikroprocesoru, velikost chyby při užití konkrétní metody a v neposlední řadě rychlost zpracování mikroprocesorem. Potenciometr s exponenciální závislostí v tomto případě simuluje obecný nelineární systém, který je za pomoci aproximačních funkcí predikován. V průmyslu se takovýchto systémů využívá hojně pro predikaci určitých nelineárních charakteristik a následné přizpůsobení systému těmto jevům.

Naměřené a vypočítané hodnoty jsou uvedeny v tabulkách a v grafech spolu s výpočty náhradních funkci funkce skutečné. Měření je do jisté míry zatíženo chybami měření, tj. přesností použitého voltmetru, přesností AD převodníku použitého mikroprocesoru PIC a především přesností zpracování nelineárního průběhu charakteristiky potenciometru.

1 Mikrokontroléry PIC

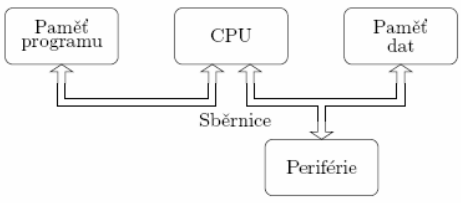
Mikrokontroléry PIC jsou programovatelná zařízení, která jsou velice rozšířená a hojně využívána díky své univerzálnosti a především díky své nízké ceně. Celý systém mikrokontroléru je totiž uzavřen v jednom pouzdře spolu se všemi důležitými komponentami, které daný jednočip tvoří. Jedná se o procesor, paměť, vstupně/výstupní porty, řadiče přerušení a dále bývají součástí A/D a D/A převodníky, komparátory, řadiče vnějších rozšiřujících pamětí apod.

1.1 Základní popis

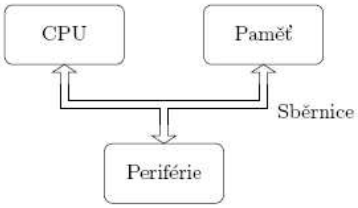


Obr. 1.1Diagram mikrokontroléru PIC24HJ32GP202 [9]

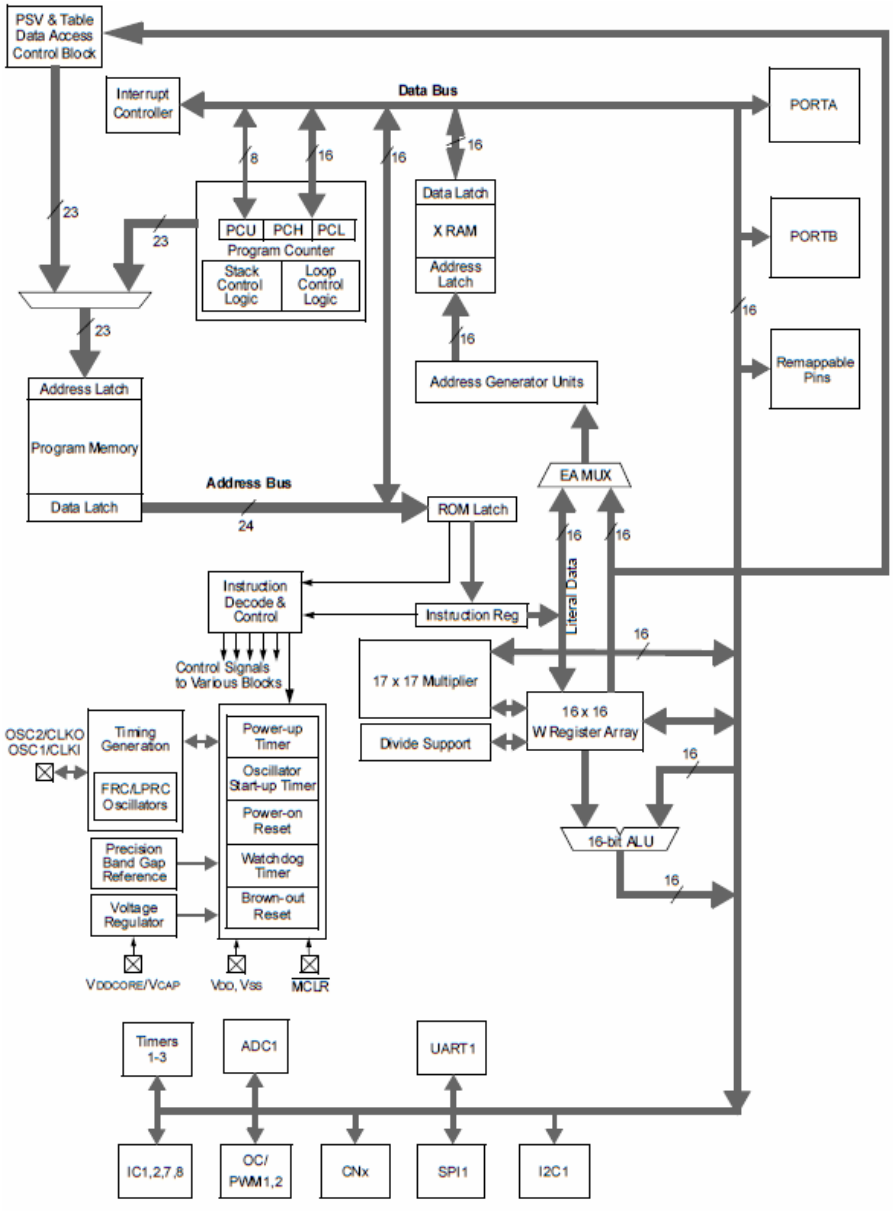
Procesor mikrokontroléru běží na taktovací frekvenci dané oscilátorem, jenž bývá tvořen buď R-C článkem nebo krystalem. Podle délky slova je pak určeno o jaký procesor se jedná, zdali 4, 8, 16 nebo 32-bitový procesor. Architektura procesorů pro PIC je založena na architektuře RISC, což znamená, že procesor pracuje s omezeným množstvím instrukcí (35 instrukcí) pevné délky (tzn. délka v bitech všech instrukcí je stejná). Instrukční sada obsahuje především jednoduché instrukce. Instrukce jsou pevné délky a doba provedení jedné instrukce se nazývá cyklus. Paměť je navržena dle Harvardské architektury (viz obr. 1.2), což znamená, že je rozdělena na paměť programu a paměť dat. Opakem této architektury je koncepce Von Neumanna (viz obr. 1.3), tj. sdílená paměť jak pro program, tak i pro data. Programová paměť a datová paměť nemají stejně dlouhé datové slovo.



Obr. 1.2 Harvardská koncepce



Obr. 1.3 Von Neumannova koncepce



Obr. 1.4 Blokový diagram mikrokontroléru PIC24HJ32GP202 [9]

Jednočipy jsou určeny pro nejrůznější řídicí a kontrolní úlohy v mnoha průmyslových oborech. Užívají se např. pro realizaci různých měřicích a řídicích systému apod. Díky své univerzálnosti, velikosti, nízké ceně a malé spotřebě nacházejí své uplatnění ve velkém množství aplikací. Mikrokontroléry lze nalézt v mnoha současných elektronických zařízeních. Čerpáno z [7].

1.2 Organizace paměti

1.2.1 Organizace programové paměti

Mikroprocesor PIC24HJ32GP202 je vybaven 23-bitovým programovým čítačem schopným adresovat 32kB prostoru v programové paměti. PIC24HJ32GP202 má fyzicky implementovanou paměť programu o velikosti 2kB. Organizace datové paměti

Paměť dat je rozdělena do dvou bank, které tvoří Speciální Funkční Registry (SFR) a Registry pro všeobecné použití (GPR).

Nastavením bitu RP0 ve STATUS registru se určí, do které banky bude nastaven přístup, tj. umožněno čtení či zápis.

Nastavení bitu RP0:

0 → Zvolena Banka 0 pro přístup,

1 → Zvolena Banka 1 pro přístup.

GPR

V „Bance 0“ je pro GPR vyhrazeno 96 bajtů a v „Bance 1“ 32 bajtů, tady celkem 128 bajtů. Každý registr je přístupný přímo či nepřímo skrz FSR.

SFR

Prvních 32 bajtů v obou bankách tvoří SFR (v Bance 0: 00h – 1Fh a v Bance 1: 80h – 9Fh). SFR jsou registry, které využívají především procesor a periferní funkce pro kontrolu požadované operace od nějakého zařízení, např. komparátoru či AD převodníku.

STATUS registr

Status registr obsahuje bity aritmetických příznaků ALU, bity pro nastavení banky paměti dat a bity pro příznaky resetu. Status registr může být cílovým registrem pro vykonání instrukce programu stejně tak jako veškeré ostatní registry paměti RWM (RAM). Pokud je ale STATUS registr cílovým registrem pro uložení výsledků jakékoli operace, která má za následek ovlivnění příznaků Z, DC nebo C, pak je zápis do těchto bitů zakázán. Mimoto bity příznaků resetů $\overline{TO}$, $\overline{PD}$ nelze programově ovlivnit, nemůžeme tedy do nich zapisovat. Proto výsledek instrukce jejímž operandem je právě registr STATUS zapíšeme do registru W (střadače). Např. instrukce CLRF STATUS vymaže horní tři bity a bit Z nastaví na hodnotu log. 1. Následný obsah STATUS registr je „000 n n1nn“, kde n = nezměněný stav.

Pro nastavení či mazaní bitů STATUS registru by měl uživatel používat výhradně instrukce BCF, BSF, SWAPF a MOVWF, které nemají vliv na nastavení příznakových bitů tohoto registru. Čerpáno z [9].

OPTION registr

Z OPTION registru lze bez omezení číst a stejně tak bez omezeni do něj i zapisovat. Tento registr obsahuje různé kontrolní bity sloužící k nastavení

čítače/časovače0, programovatelné předděličky, vnějšího přerušovacího vstupu RA2/INT, a pull-up rezistorů brány A.

INTCON Register

Z INTCON registru lze bez omezení číst a stejně tak bez omezeni do něj i zapisovat. Obsahuje bity určené pro práci s přerušením.

1.2.2 Registry PCL a PCLATH

Programový čítač (PC) obsahuje adresu instrukce, která se má vykonat v následujícím cyklu. Procesory PIC používají tzv. zřetězení – během výkonu instrukce dochází k načítání instrukce následující (ležící na adrese PC+1). PC má celkem 23 bitů a je tvořen dvěma registry PCL a PCH. PCL obsahuje, nižší bajt programového čítače a lze ho číst a zapisovat do něj. PCH obsahuje horních 5 bitů PC a není přístupný pro čtení ani zápis. Instrukce, které mění skokově obsah PC vyžadují dva instrukční cykly. Dochází k novému načítaní následující instrukce, a již načtená (PC+1) je ignorována (vykoná se NOP - No Operation).

PCH lze měnit pouze pomocí záchytného registru PCLATH. Jakákoli instrukce, ve které je cílovým operandem PCL automaticky provede zápis z PCLATH do PCH. Výsledkem je skok v rámci celé paměti programu.

Při resetu se PC vynuluje. Vpravo je příklad načtení hodnoty z PCLATH registru do horních pěti bitů PC, tj. do PCH (PCLATH<4:0> → PCH). Ve spodní části obrázku je zas znázorněno, jak PC načítá v průběhu vykonávání CALL nebo GOTO instrukce (PCLATH<4:3> → PCH). Čerpáno z [1].

Zásobník (STACK)

Zásobník obsahuje návratovou hodnotu, adresu, pro návrat z podprogramu. Není součástí paměti programu ani paměti dat a nelze do zásobníkového ukazatele (STACK POINTER) ani zapisovat ani z něj nelze číst. Do zásobníku se ukládá hodnota z programového čítače (Program Counter). Zápis resp. čtení ze zásobníku provádí procesor výlučně na základě vykonání instrukcí programu. Zásobník umožňuje celkem osm úrovní vnoření (STACK LEVEL 1-8). Celkově se tedy může volat až 8 podprogramů vnořených do sebe bez návratové instrukce. Zásobník má šířku 23 bitů. Čerpáno z [9].

1.3 Vstupně/výstupní brány

Mikropočítač má 2 brány, PORTA a PORTB. Některé vývody PIC jsou použity nejen jako univerzální digitální brány, ale je možné jejich použití pro činnost některého z periferních zařízení PIC. Obecně platí, že pokud je vývod PIC použit pro funkci periferního obvodu, nelze jej užít jako univerzální číslicový vstup/výstup.

PORTA je 5-ti bitová obousměrná datová brána. Pro přístup a užívání PORTA brány slouží záchytný registr TRISA, který obsahuje 5 funkčních bitů. Pro použití PORTA brány jako vstupní, je potřeba nastavit jednotlivé bity TRISA registru do stavu log.1 (stav vysoké impedance), pokud je bit vymazán, jemu odpovídající pin je nastaven ve výstupním režimu (tím se objeví hodnota registru PORTA na vývodech brány PortA).

PORTB je 16-ti bitová obousměrná datová brána. Piny brány lze nastavit buď jako vstupně/výstupní digitální bránu nebo jako analogové vstupy do A/D převodníku nebo jako komparátor. Čerpáno z [9].

1.4 Čítače/Časovače

1.4.1 TIMER0

Modul čítače/časovače má následující základní vlastnosti:

- Šířka pracovního registru TMR0 je 8 bitů,
- Z registru čítače/časovače TMR0 lze číst i do něj zapisovat,
- Volitelný zdroj hodinového signálu, vnější a vnitřní,
- Volitelná aktivní hrana vnějšího hodinového signálu,
- 8-bitová programovatelná předdělička,
- Přerušení přetečením čítače/časovače z FFh to 00h.

1.4.2 TIMER1

Modul čítače/časovače má následující základní vlastnosti:

- Šířka pracovního registru TMR1 je 16 bitů (párové registry TMR1H a TMR1L),
- Volitelný zdroj hodinového signálu, vnější a vnitřní,
- 3-bitová programovatelná předdělička,
- Volitelný LP oscilátor,
- Synchronní nebo asynchronní operace,
- Přerušení přetečením čítače/časovače z FFFFh to 00000h,
- „Wake-up“ při přetečení (externí hodiny, asynchronní režim pouze),
- Časová základna pro funkce zachytávání/ porovnávání.

1.4.3 TIMER2

V Timer2 modulu je 8-bit časovač s těmito rysy:

- 8-bitový časovač (TMR2),
- 8-bitový registr PR2,
- Přerušení TMR2 v případě rovnosti s hodnotou v registru PR2,
- Programovatelná předdělička (1:1, 1:4, 1:16),
- Programovatelná předdělička (1:1 až 1:16).

1.5 Analogově-digitální převodník

Analogově-digitální převodník (ADC) umožňuje konverzi analogového vstupního signálu na 10-bitové nebo 12-bitové binární zastoupení tohoto signálu. Tento přístroj používá analogové vstupy, které jsou multiplexovány do jednoho vzorku. Ze vzorku na převodníku s postupnou aproximací se vygeneruje 10-bitová nebo 12-bitová binární hodnota. Výsledek převodu, tj. digitální hodnota, je zapsána do registrů (ADRESL a ADRESH). Jako referenční napětí AD převodníku je využito buď napájecího napětí VDD nebo externí napětí přivedené na tomu určeném pinu.

Převodník může vygenerovat přerušení po dokončení konverze analogového signálu na digitální. Toto přerušení může být použito pro „probuzení“ (Wake-up) mikrokontroléru z klidového režimu „Sleep“.

Modul AD převodníku má celkem 5 registrů:

- ADRESH je registr pro horní část výsledku,
- ADRESL je registr pro dolní část výsledku,
- ADCON0 je první řídící registr,
- ADCON1 je druhý řídící registr,
- ANSEL je registr pro konfiguraci analogových vstupů.

2 Aproximace funkce

Aproximace funkcí slouží k náhradě jedné funkce za funkci jinou, která je jednodušší pro následné zpracování, se kterou se lépe počítá. Je několik možností této náhrady.

Na vstupu mikrokontroléru bude potenciometr s nelineární, resp. exponenciální závislostí, tzn. že průběh nárůstu napětí na potenciometru není lineární vůči změně polohy, tj. vůči pootočení jezdce po odporové dráze. Potenciometr zde simuluje nelineární systém. Potenciometr má mít odstupňovanou dráhu a mikrokontrolér bude odečítat změny napětí po určitých skocích. Získá se soubor několika hodnot, uzlových bodů, charakteristiky daného potenciometru. Cílem bude dopočítat celou charakteristiku pomocí vybrané aproximační funkce, která co nejvěrněji bude popisovat průběh charakteristiky skutečné, výstupem tedy bude rovnice regrese, konkrétně logaritmické funkce, která bude náhradou funkce skutečné, jejíž uzlové body budou známy. Při výběru konkrétní aproximační funkce se musí počítat s jistými chybami dané např. ne zcela ideálním průběhem daného potenciometru a chybou konverze AD převodníku. Dále je nutné při výběru zohlednit náročnost implementace posuzované metody do mikrokontroléru, náročnost na paměťový prostor vybrané funkce v mirokontroléru, náročnost metody samotné, tj. jejího výpočtu a tedy následnou rychlost na zpracování mikrokontrolérem.

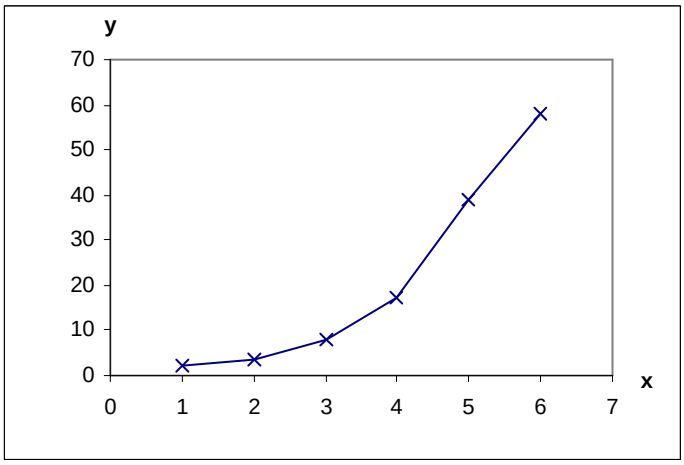
2.1 Rozdělení aproximačních funkcí

Aproximační funkce se dají rozdělit do těchto dvou základních skupin:

- Interpolace,
- Metoda nejmenších čtverců.

Interpolace je taková aproximace, při níž náhradní funkce $\varphi(x)$, funkce původní $f(x)$, nabývá v zadaných bodech (námi změřených uzlových bodech) x_i předepsaných, daných hodnot, $y_i = f(x_i)$. Interpolace se dále dělí na:

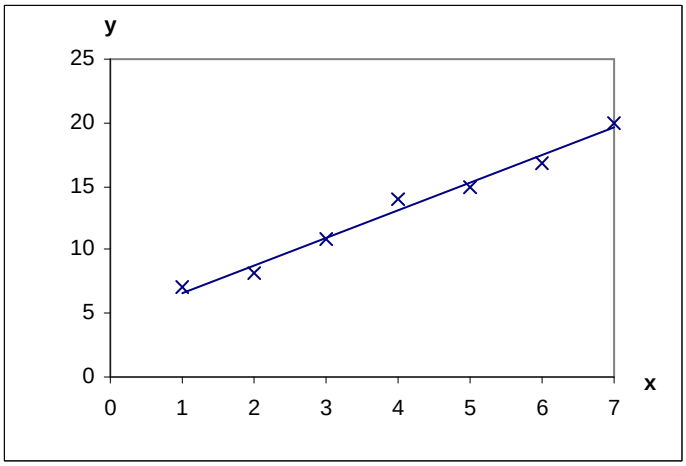
- interpolace polynomem,
- interpolace funkcí více proměnných,
- interpolace splajny.



Obr. 2.1 Příklad interpolace lineárním splajnem

Metoda nejmenších čtverců je taková aproximace, při níž náhradní funkci φ(x) prokládáme mezi uzlovými body funkce původní f(x). U této metody bývá charakteristické, že funkce φ(x) uzlovými body funkce f(x) neprochází. Tato metoda se dá rozdělit na:

- lineární regresi (aproximace přímkou),
- polynomická regrese (aproximace polynomem),
- kvadratická regrese (aproximace parabolou).



Obr. 2.2 Příklad lineární regrese

Na základě výše uvedeného je vhodnější výběr metody nejmenších čtverců a to z toho důvodu že při interpolaci musí náhradní funkce procházet uzlovými body funkce původní a náhradní funkce by byla zatížena chybami měření narozdíl od metody nejmenších čtverců, která se snaží redukovat velikost chyb a to tak, že se mezi uzlovými body původní funkce prokládá jistým způsobem, jež umožňuje co nejvíce minimalizovat velikost chyb. Čerpáno z [4].

2.2 Metoda nejmenších čtverců

Metoda nejmenších čtverců je tedy pro požadavky výpočtu náhradní funkce φ(x) nejvhodnější. Dalším důvodem proč je vhodnější metoda nejmenších čtverců narozdíl od ostatních aproximačních metod je konkrétní představa o povaze funkce, jejíž uzlové hodnoty jsou měřeny a tudíž známy , tzn. že pokud se jedná o průběh exponenciální, tak budeme hledat mezi všemi funkcemi tohoto známého typu takovou, která prochází k zadaným bodům v jistém smyslu nejblíže. Čerpáno z [4].

2.2.1 Lineární regrese (aproximace přímkou)

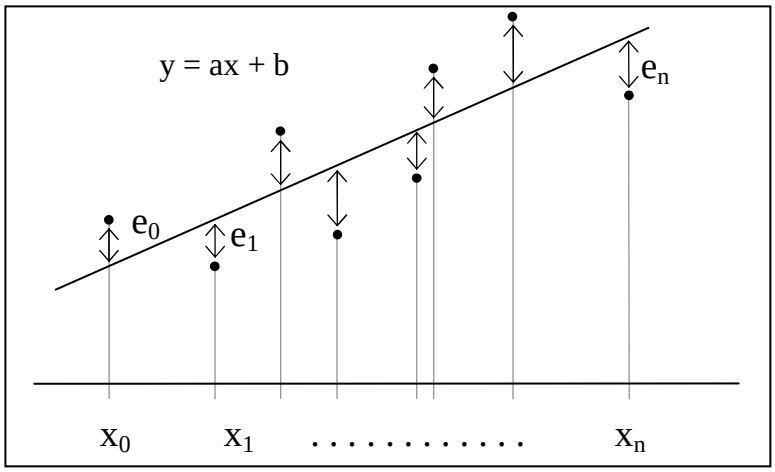
Jsou známy body charakteristiky potenciometru x_1, x_2,…., x_n, tedy x_i, kde i=0,1,…,n a funkční hodnoty v nich y_i. Cílem bude nalézt náhradní funkci funkce f(x) a to funkci φ(x), tedy v tomto případě rovnici přímky

$$\varphi(x) = y = ax + b\,, \tag{2.1}$$

která bude mezi uzlovými body $[x_i,y_i]$, kde i=0,1,..,n, „co nejlépe“ procházet. Tato aproximace má potom chybu e_i v každém i-tém bodě, tedy

$$e_i = y_i - y(x_i) = y_i - ax_i - b\,, \tag{2.2}$$

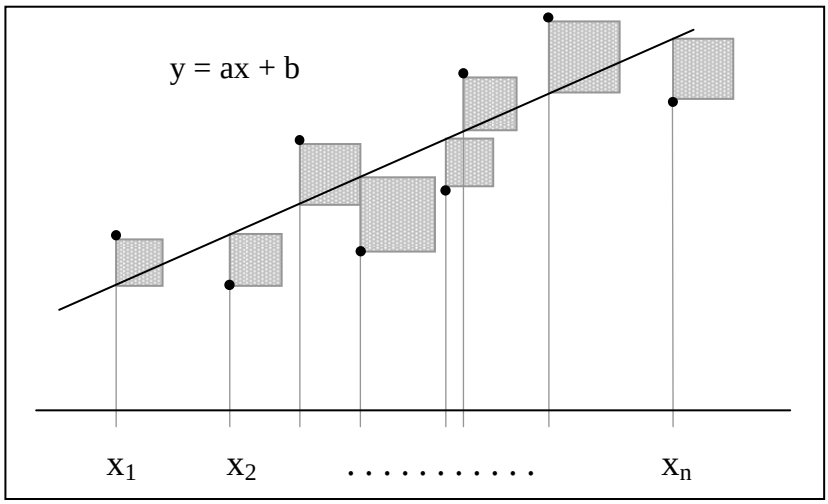
viz obrázek 2.3. Jelikož body $[x_i,y_i]$ jsou dány, chyba závisí pouze na koeficientech přímky *a* a *b*.



Obr. 2.3 Odchylky e_i

Z výše uvedeného se jeví jako vhodné kritérium pro určení onoho „co nejlepšího“ procházení, proložení přímky mezi body tak, aby součet druhých mocnin (neboli čtverců) chyb v jednotlivých bodech byl co nejmenší, viz obr. 2.4. Tento součet se značí ρ2. Cílem tedy bude minimalizace funkce

$$\rho^2(a,b) = (y_0 - ax_0 - b)^2 + (y_1 - ax_1 - b)^2 + ... + (y_n - ax_n - b)^2 = \\ = \sum_{i=0}^{n} (y_i - ax_i - b)^2. \tag{2.3}$$



Obr. 2.4 Hledání rovnice, pro niž je součet obsahů čtverců nejmenší

Veličina ρ^2 se nazývá kvadratickou odchylkou. Potřebná podmínka pro to, aby $\rho^2(a,b)$ nabývala minima, je splnění rovnic

$$\frac{\partial(\rho^2)}{\partial a} = 0 \quad \text{a} \quad \frac{\partial(\rho^2)}{\partial b} = 0. \tag{2.4}$$

Výpočet parciální derivace podle *b*:

$$\frac{\partial(\rho^2)}{\partial b} = 2(y_0 - ax_0 - b)(-1) + 2(y_1 - ax_1 - b)(-1) + ... + 2(y_n - ax_n - b)(-1) = \\ = -2\left[(y_0 - ax_0 - b) + (y_1 - ax_1 - b) + ... + (y_n - ax_n - b)\right] = \\ = -2\left[(y_0 + y_1 + ... + y_n) - a(x_0 + x_1 + ... + x_n) - b(1 + 1 + ... + 1)\right] = \\ = -2\left[\sum_{i=0}^{n} y_i - a\sum_{i=0}^{n} x_i - b(n+1)\right]. \tag{2.5}$$

Výpočet parciální derivace podle *a*:

$$\frac{\partial(\rho^2)}{\partial a} = \sum_{i=0}^{n} 2(y_i - ax_i - b)(-x_i) = -2\sum_{i=0}^{n}(x_i y_i - ax_i^2 - bx_i) = \\ = -2(\sum_{i=0}^{n} x_i y_i - a\sum_{i=0}^{n} x_i^2 - \sum_{i=0}^{n} x_i) \ . \tag{2.6}$$

Po úpravě rovnic (2.5) a (2.6), kdy jsou položeny rovny 0 a po odstranění koeficientů -2, jsou výsledkem úprav tzv. normální rovnice s neznámými koeficienty *a* a *b*

$$a\sum_{i=0}^{n} x_i + b(n+1) = \sum_{i=0}^{n} y_i \ , \tag{2.7}$$

$$a\sum_{i=0}^{n} x_i^2 + b\sum_{i=0}^{n} x_i = \sum_{i=0}^{n} x_i y_i \ . \tag{2.8}$$

Nyní se mohou pomocí lineární regrese aproximovat i složitější závislosti. Lineární regresi lze tedy použít i na výpočet logaritmickou závislost z naměřených hodnot charakteristiky potenciometru, kdy rovnice této regrese je:

$$\varphi(x) = y(x) = B\ln x - A. \tag{2.9}$$

Pro výpočet logaritmické závislosti je ale potřeba nejprve upravit nezávislou proměnnou *x*, tak aby bylo možnou použít rovnic pro výpočet lineární regrese pro případ ne lineární závislosti, ale logaritmické, a to tak, že namísto proměnné *x* se do vzorců (2.7) a (2.8) dosadí přirozený logaritmus nezávislé proměnné *x*.

Vyjádření koeficientů *a* a *b* z rovnic (2.7) a (2.8) po úpravě :

$$a = A = \frac{(n+1)\left(\sum_{i=0}^{n} y_i \ln x_i\right) - \left(\sum_{i=0}^{n} \ln x_i\right)\left(\sum_{i=0}^{n} y_i\right)}{(n+1)\sum_{i=0}^{n}(\ln x)^2 - \left(\sum_{i=0}^{n} \ln x_i\right)^2} \ , \tag{2.10}$$

$$b = B = \frac{\left(\sum_{i=0}^{n} (\ln x_i)^2\right)\left(\sum_{i=0}^{n} y_i\right) - \left(\sum_{i=0}^{n} \ln x_i\right)\left(\sum_{i=0}^{n} y_i \ln x_i\right)}{(n+1)\sum_{i=0}^{n} (\ln x)^2 - \left(\sum_{i=0}^{n} \ln x_i\right)^2} \quad . \tag{2.11}$$

Složitější případ nastane, pokud bude mít funkční závislost dvou veličin obecný tvar. Tehdy se použije polynomická regrese (polynomiální aproximace), kdy neznámou veličinu aproximujeme polynomem *k*-tého řádu. Čerpáno z [4]

2.2.2 Polynomická regrese (aproximace polynomem)

Lineární regrese je aproximace funkce polynomem 1. stupně. Polynomická regrese je obecně aproximace polynomem stupně *m*, který má funkci

$$\mathrm{P_m(x) = c_0 + c_1 x + \ldots + c_m x^m}, \tag{2.12}$$

kde m je stupeň polynomu.

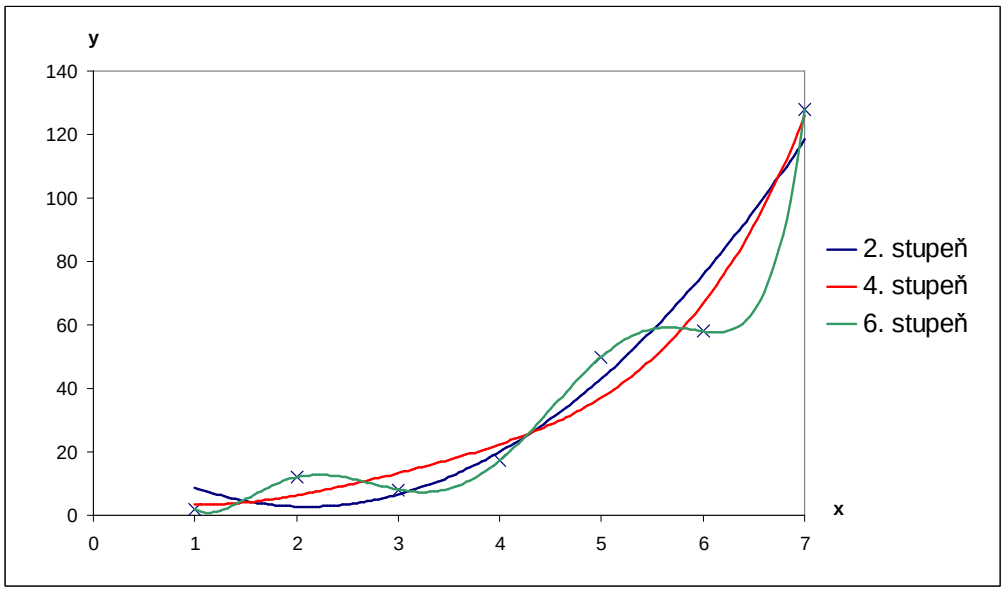
Postup řešení této funkce je pak totožný s postupem v případě přímky, kdy je nutno vypočítat parciální derivace kvadratické odchylky

$$\begin{aligned}
&\rho^2(c_0, c_1, ..., c_m) = \\
&= (y_0 - c_0 - c_1 x_0 - ... - c_m x_0^m)^2 + (y_1 - c_0 - c_1 x_1 - ... - c_m x_1^m)^2 + \\
&+ ..... + (y_n - c_0 - c_1 x_n - ... - c_m x_n^m)^2 = \\
&= \sum_{i=0}^{n} (y_i - c_0 - c_1 x_i - ... - c_m x_i^m)^2 \quad ,
\end{aligned} \tag{2.13}$$

z nichž následně vzejdou rovnice tvaru

$$\begin{aligned}
&c_0(n+1) + c_1 \sum_{i=0}^{n} x_i + ... + c_m \sum_{i=0}^{n} x_i^m = \sum_{i=0}^{n} y_i \quad , \\
&c_0 \sum_{i=0}^{n} x_i + c_1 \sum_{i=0}^{n} x_i^2 + ... + c_m \sum_{i=0}^{n} x_i^{m+1} = \sum_{i=0}^{n} x_i y_i \quad , \\
&\qquad\qquad\qquad\qquad \vdots \\
&c_0 \sum_{i=0}^{n} x_i^m + c_1 \sum_{i=0}^{n} x_i^{m+1} + ... + c_m \sum_{i=0}^{n} x_i^{2m} = \sum_{i=0}^{n} x_i^m y_i \quad .
\end{aligned} \tag{2.14}$$

Polynomická regrese je taktéž vhodná pro implementaci do mikrokontroléru. Je však nutné vhodně zvolit stupeň polynomu (viz obr.2.5). Pokud by byl stupeň polynomu příliš malý nebo naopak příliš velký byla by potom chyba aproximace značná, což je nežádoucí. Pro vhodný výběr stupně polynomu je nutné zohlednit také chyby, kterými jsou ovlivněny naměřené uzlové body, aby se pak aproximací dané chyby nezvětšovaly, ale naopak co nejvíce redukovaly.



Obr. 2.5 Aproximace polynomem

2.2.3 Kvadratická regrese (aproximace parabolou)

Aproximace parabolou se řeší obdobným způsobem, jako se řeší aproximace přímkou či polynomem. Pro naměřené uzlové body $[x_i,y_i]$, kde i=0,1,..,n, by se hledala aproximační funkce a to parabola, která má rovnici ve tvaru

$$y = ax^2 + bx + c \ , \tag{2.15}$$

pro niž je minimální kvadratická odchylka rovna

$$\begin{aligned} &\rho^2(a,b,c) = (y_0 - ax_0^2 - bx_0 - c)^2 + (y_1 - ax_1^2 - bx_1 - c)^2 + ... + \\ &+ (y_n - ax_n^2 - bx_n - c)^2 = \\ &= \sum_{i=0}^{n} (y_i - ax_i^2 - bx_i - c)^2 . \end{aligned} \tag{2.16}$$

Dále je nutno vyřešit rovnice

$$\frac{\partial(\rho^2)}{\partial a} = 0 \ , \qquad \frac{\partial(\rho^2)}{\partial b} = 0 \quad \text{a} \qquad \frac{\partial(\rho^2)}{\partial c} = 0 \ . \tag{2.17}$$

Výpočet parciální derivace podle a :

$$\begin{aligned} &\frac{\partial(\rho^2)}{\partial a} = 2(y_0 - ax_0^2 - bx_0 - c)(-x_0^2) + 2(y_1 - ax_1^2 - bx_1 - c)(-x_n^2) + ... + \\ &+ 2(y_n - ax_n^2 - bx_n - c)(-x_n^2) = \\ &= -2\left[\begin{matrix}(x_0^2 y_0 - ax_0^4 - bx_0^3 - cx_0^2) + (x_1^2 y_1 - ax_1^4 - bx_1^3 - cx_1^2) + ... + \\ + (x_n^2 y_n - ax_n^4 - bx_n^3 - cx_n^2)\end{matrix}\right] = \\ &= -2\left[\sum_{i=0}^{n} x_n^2 y_n - a\sum_{i=0}^{n} x_n^4 - b\sum_{i=0}^{n} x_n^3 - c\sum_{i=0}^{n} x_n^2\right] . \end{aligned} \tag{2.18}$$

Po vypočtu ostatních parciálních derivací podle koeficientu *b* a *c* je výsledkem po patřičné úpravě následující soustava rovnic

$$\begin{aligned} a\sum_{i=0}^{n} x_i^4 + b\sum_{i=0}^{n} x_i^3 + c\sum_{i=0}^{n} x_i^2 &= \sum_{i=0}^{n} y_i\, x_i^2 \quad , \\ a\sum_{i=0}^{n} x_i^3 + b\sum_{i=0}^{n} x_i^2 + c\sum_{i=0}^{n} x_i &= \sum_{i=0}^{n} y_i\, x_i \quad , \\ a\sum_{i=0}^{n} x_i^2 + b\sum_{i=0}^{n} x_i + c(n+1) &= \sum_{i=0}^{n} y_i \quad . \end{aligned} \tag{2.19}$$

Soustava rovnic (2.19) se pak může přepsat do maticového tvaru

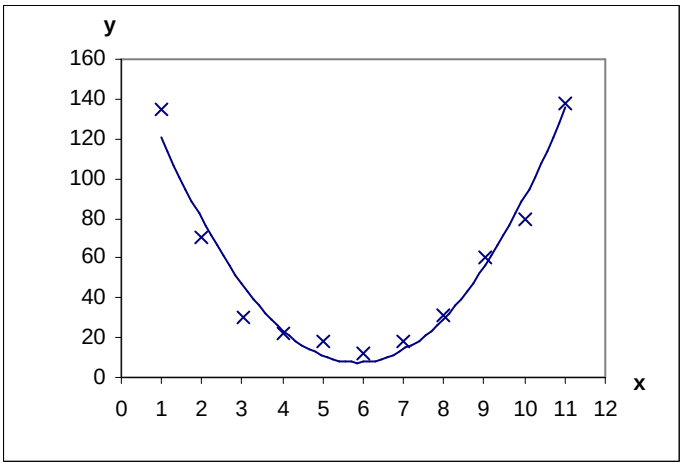
$$Z^T Z k = Z^T y, \tag{2.20}$$

kde

$$Z = \begin{pmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{pmatrix}, \qquad k = \begin{pmatrix} a \\ b \\ c \end{pmatrix}, \qquad y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

Ze soustavy (2.20) se pak může vyjádřit neznámý vektor *k* (konstanty *a*, *b*, *c*) jako

$$k = (Z^T Z)^{-1} Z^T y \ .$$



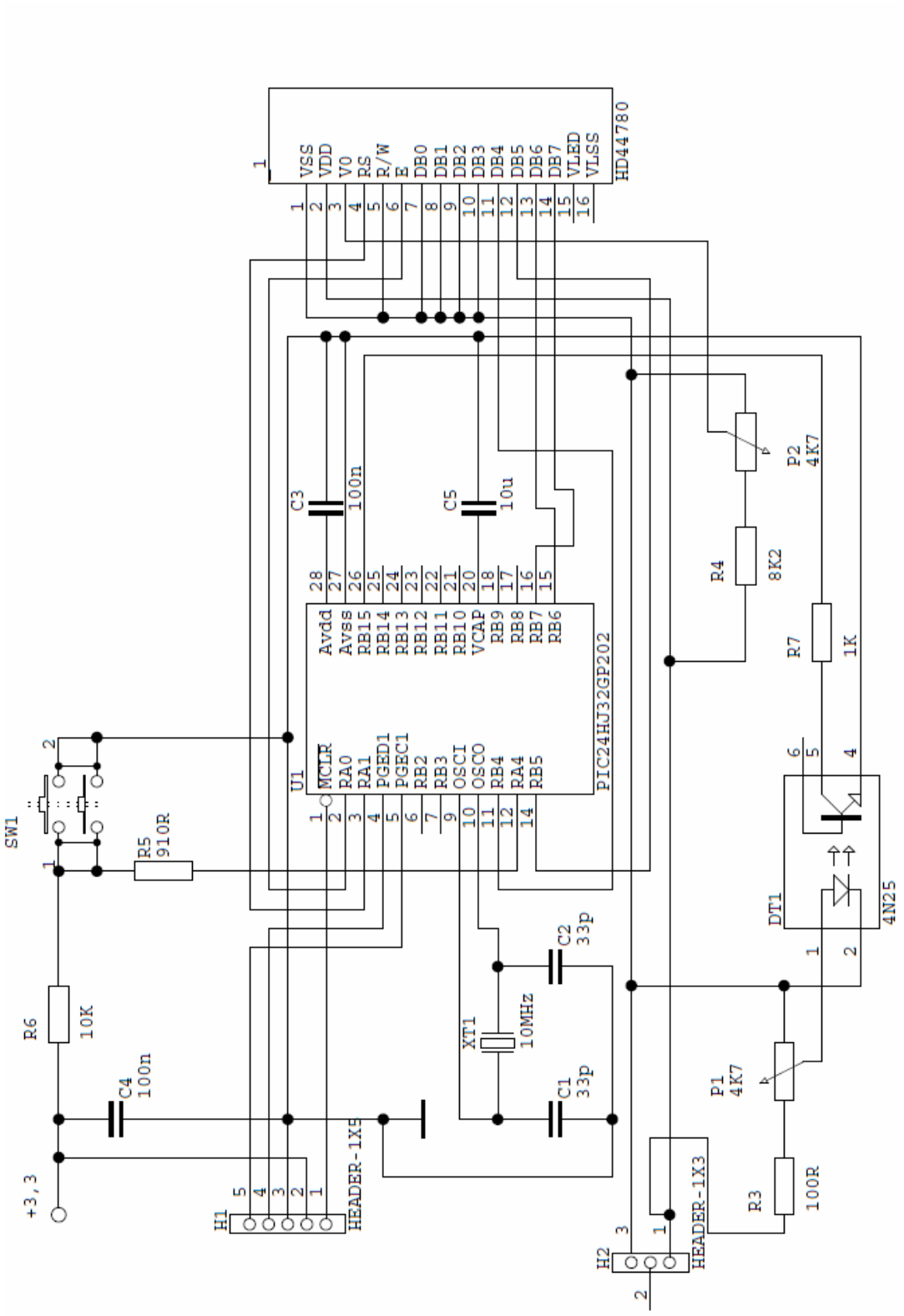
Obr. 2.6 Aproximace parabolou

2.2.4 Srovnání metod nejmenších čtverců

Z hlediska složitosti implementace jednotlivých metod do mikroprocesoru není mezi metodami výrazný rozdíl, tzn. že implementace by neměla být příliš složitá.

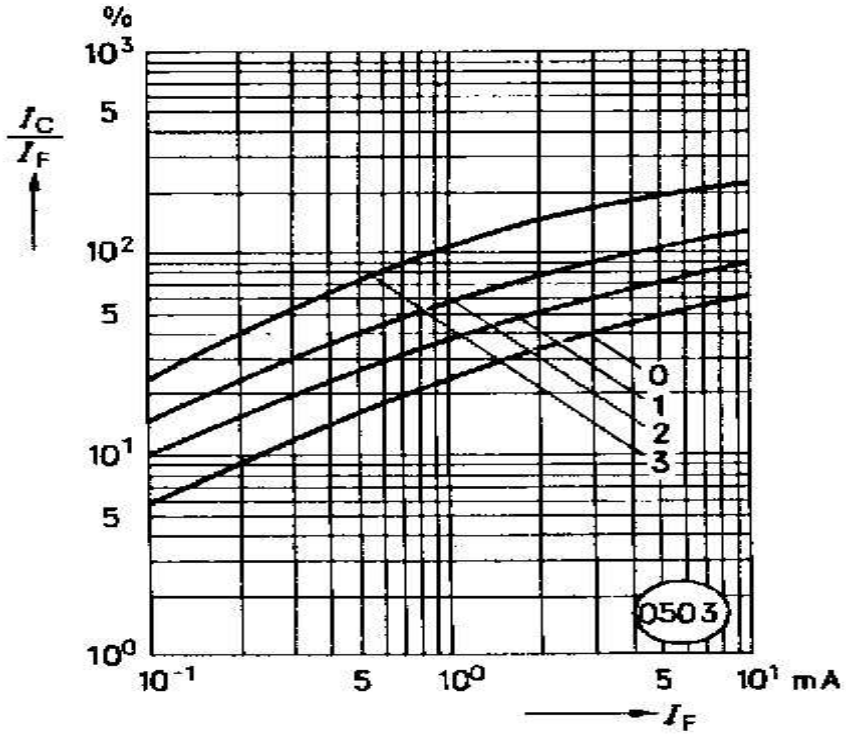
Z hlediska velikosti chyb při aproximaci se z uvedených metod jeví pro implementaci jako nejvýhodnější užití aproximace přímkou. Jelikož je závislost potenciometru exponenciální, tak je výhodné upřednostnit právě lineární regresi. U lineární regrese se po zlogaritmování naměřených hodnot lehce dopočítá rovnice regrese, konkrétně rovnice logaritmické funkce. Regrese parabolou není vhodná, jelikož rovnicí regrese je parabola a to pro řešení této práce ztrácí význam. Regrese polynomem by byla možná, avšak by byl problém s vhodnou volbou stupně polynomu. Poněvadž nejsou zpočátku měření známy všechny uzlové body charakteristiky potenciometru, je volba stupně polynomu složitá. Volba stupně by tak byla pouze intuitivní a chyba by mohla být příliš velká a tudíž by rozhodně nebyla zanedbatelná.

3 Schéma zapojení a postup měření



Obr. 7 Schéma zapojení pro měření

K mikroprocesoru je připojen nelineární potenciometr (logaritmický), jenž představuje nějaký nelineární systém, který se má proměřovat. Od mikroprocesoru je galvanický oddělen. Je tomu kvůli rozdílu napěťových potenciálů mezi obvodem mikroprocesoru a obvodem s měřeným potenciometrem. Jelikož převodní charakteristika optočlenu není lineární, je nutno snažit se obvod nastavit tak, aby pracovní bod byl v části charakteristiky, která se nejvíce blíží linearitě (viz obr.8).



Obr. 8 Převodní charakteristika IC/IF = f (IF) při TA = 25°C, VCE = 5.0 V. [1]

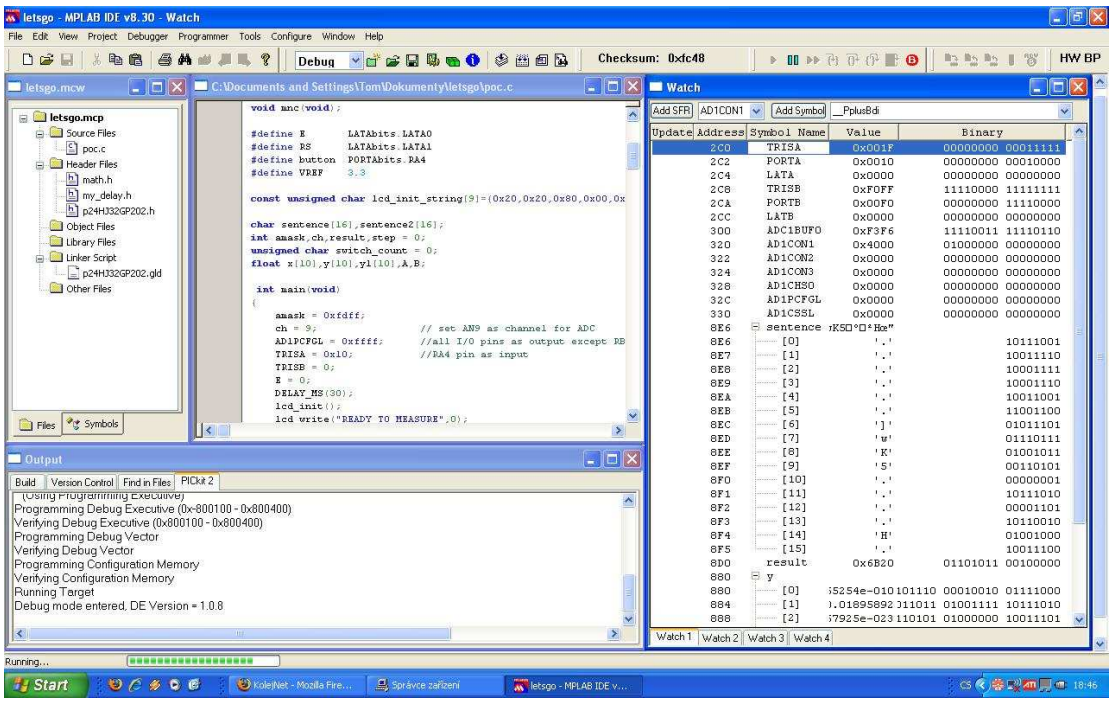
Výstup optočlenu je pak přiveden na vstup portu B mikroprocesoru, pin RB15, který je nastaven jako analogový vstup 12-bitového A/D převodníku. Výstup měření bude zobrazován na dvouřádkovém LCD displeji, jenž je řízen řadičem HD44780. Komunikace PIC a řadičem displeje je 4-bitová (u řadiče displeje vstupy DB4-DB7). Oproti komunikaci 8-bitové je 4-bitová komunikace 2x pomalejší, jelikož se pošlou nejprve vyšší čtyři bity a poté nižší čtyři bity, kdežto u 8-bitové se pošle všech osm bitů najednou. Protože je rychlost přenosu dat z mikroprocesoru do řadiče dostatečná, není potřeba využívat dalších 4 výstupů PIC, které by mohly být případně využity pro jiné účely.

Na pin RA4 je připojeno tlačítko, pomocí kterého se bude uskutečňovat měření charakteristiky potenciometru. Tlačítko je zapojeno tak, že při nestisknutém stavu mikrokontrolér indikuje na vstupu log. 1, do doby kdy se tlačítko stiskne. V momentu stisku tlačítka a po dobu jeho trvání je na vstupu mikrokontroléru log. 0.

Programování a samotná komunikace s mikrokontrolérem bude probíhat přes vstupy PGED1, do něhož budou vstupovat data, a PGEC1, do něhož PICKit2 posílá hodinový signál, který umožňuje synchronizovat komunikaci mezi počítačem a mikrokotrolérem PIC24HJ32GP.

4 Vývoj a ladění zdrojového kódu

Program pro obsluhu mikroprocesoru je napsán v jazyce C v prostředí MPLAB (obr.9), za použití překladače „C 30“, který je určen pro 16-bitové mikrokontroléry.



Obr. 9 Prostředí programu MPLAB v8.30

V překladači byly k dispozici základní knihovny, které se běžně v jazyce C používají. Součástí zdrojového kódu je volání knihoven pro pokročilejší matematické operace math.h, v našem případě logaritmy (příkaz logf( )), a pro práci s řetězci stdio.h (příkaz sprintf( )). Pak se dále volá knihovna pro obsluhu prodlev my_delay.h a jako nejdůležitější součást kódu pro práci s mikrokontrolérem, knihovna p24hj32gp202.h, kde jsou definovány všechny vstupy/výstupy, čítače, A/D převodníky, velikosti paměti, atd.

4.1 Základní nastavení mikroprocesoru

Dále následuje základní nastavení PIC, ve kterém se definuje hodinový signál, tzv. „Watchdog Timer“ a programovací vstupy. Jako zdroj základního hodinového signálu je nastaven krystalový oscilátor s taktem 10MHz, na který se mikrokontrolér přepne s vnitřního oscilátoru po jeho zapnutí. WDT je vypnut. Jako programovací vstupy jsou nastaveny vstupy PGD1, tedy PGED1 pro data a PGEC1 pro hodinový signál, který synchronizuje komunikaci mezi mikrokontrolérem a PICKitem (přesné nastavení konfiguračních bitů mikrokontroléru je vidět na obr. 14) .

Address	Value	Category	Setting
F80000	000F	Boot Segment Write Protect	Boot Segment may be written
		Boot Segment Program Flash Code Protection	No Boot Segment
F80004	0007	General Code Segment Write Protect	General Segment may be written
		General Segment Code Protection	Disabled
F80006	FFFA	Oscillator Mode	Primary Oscillator (XT, HS, EC)
		Internal External Switch Over Mode	Start up with FRC, then switch
F80008	FF59	Primary Oscillator Source	XT Oscillator Mode
		OSCO/OSC2 Pin Function	OSCO pin has digital I/O function
		Peripheral Pin Select Configuration	Allow Multiple Re-configurations
		Clock Switching and Monitor	Sw Enabled, Mon Disabled
F8000A	FF7F	Watchdog Timer Postscaler	1:32,768
		WDT Prescaler	1:128
		Watchdog Timer Window	Non-Window mode
		Watchdog Timer Enable	Disable
F8000C	00F7	POR Timer Value	128ms
		Alternate I2C pins	I2C mapped to SDA1/SCL1
F8000E	FF7F	Comm Channel Select	Use PGC1/EMUC1 and PGD1/EMUD1
		JTAG Port Enable	Enabled

Obr. 10 Základní nastavení mikroprocesoru

4.2 Velikost a optimalizace kódu

Bylo nutno použít mikroprocesor z vyšší řady a z důvodu náročnosti programu na paměť programu. Zdrojový kód pro obsluhu mikroprocesoru zabírá 38% z celkové paměti pro program o velikosti 32kB, což odpovídá přibližně 12,8kB (obr.15). Tato vcelku veliká náročnost na paměťový prostor programu je způsoben nutností použití knihovny pro matematické operace math.h. Paměť dat je využita ze 17% z celkové velikosti 2kB, což odpovídá přibližně 360B.

Optimalizace kódu je automaticky prováděna překladačem jazyka C (C 30), kdy se kód převede do jazyka assembler a poté se teprve převede do strojového kódu, který je uložen pomocí PICKitu do paměti program mikroprocesoru PIC.

```
Program Memory  [Origin = 0x200, Length = 0x5600]

section                   address   length (PC units)   length (bytes) (dec)
-------                   -------   -----------------   --------------------
.text                       0x200             0x1d2c          0x2bc2  (11202)
.const                     0x1f2c               0x70            0xa8  (168)
.text                      0x1f9c             0x2e4           0x456  (1110)
.dinit                     0x2280               0xca           0x12f  (303)
.text                      0x234a               0x1c            0x2a  (42)
.isr                       0x2366                0x2             0x3  (3)

                     Total program memory used (bytes):          0x321c  (12828) 38%


Data Memory  [Origin = 0x800, Length = 0x800]

section                   address      alignment gaps    total length  (dec)
-------                   -------      --------------    ------------------
.nbss                       0x800                  0            0xa6  (166)
.ndata                      0x8a6                  0             0x4  (4)
.nbss                       0x8aa                  0             0x2  (2)
.data                       0x8ac                  0            0x48  (72)
.dconst                     0x8f4                  0            0x32  (50)
.data                       0x926                  0            0x34  (52)
.dconst                     0x95a                  0             0x8  (8)
.data                       0x962                  0             0x2  (2)
.dconst                     0x964                  0             0x2  (2)
.data                       0x966                  0             0x2  (2)

                    Total data memory used (bytes):          0x168  (360) 17%


Dynamic Memory Usage

region                    address                    maximum length  (dec)
------                    -------                    --------------------
heap                        0x968                           0x200  (512)
stack                       0xb68                           0x498  (1176)

                    Maximum dynamic memory (bytes):          0x698  (1688)
```

Obr. 11 Využití paměti programu a paměti dat mikroprocesoru

4.3 Popis funkce pro výpočet aproximační funkce

Funkce, pomocí které se provádí samotný výpočet aproximační funkce, se jmenuje mnc() (viz příloha A). Vzorce pro výpočet koeficientů *A* (2.9) a *B* (2.10) se rozdělily na dílčí jednodušší vzorce (i=1,2,…,n):

$$sum1 = \sum_{i=1}^{n} y_i \ln x_i , \tag{4.1}$$

$$sum2 = \sum_{i=1}^{n} \ln x_i , \tag{4.2}$$

$$sum3 = \sum_{i=1}^{n} y_i , \tag{4.3}$$

$$sum4 = \sum_{i=1}^{n} (\ln x_i)^2 , \tag{4.4}$$

$$sum5 = \sum_{i=1}^{n} x_i . \tag{4.5}$$

Následně se vypočítá čitatel koeficientu *B* (*num1*) a koeficientu *A* (*num2*):

$$num1 = n \cdot sum1 - sum2 \cdot sum3, \tag{4.6}$$

$$num2 = sum4 \cdot sum3 - sum2 \cdot sum1. \tag{4.7}$$

Společný jmenovatel koeficientů *A* a *B* :

$$den = n \cdot sum4 - (sum2)^2 . \tag{4.8}$$

Takovýmto rozdělením původních složitých vzorců na vzorce jednodušší se zápis funkce v jazyce C zjednoduší a navíc zpřehlední, jak lze vidět na následujících vztazích:

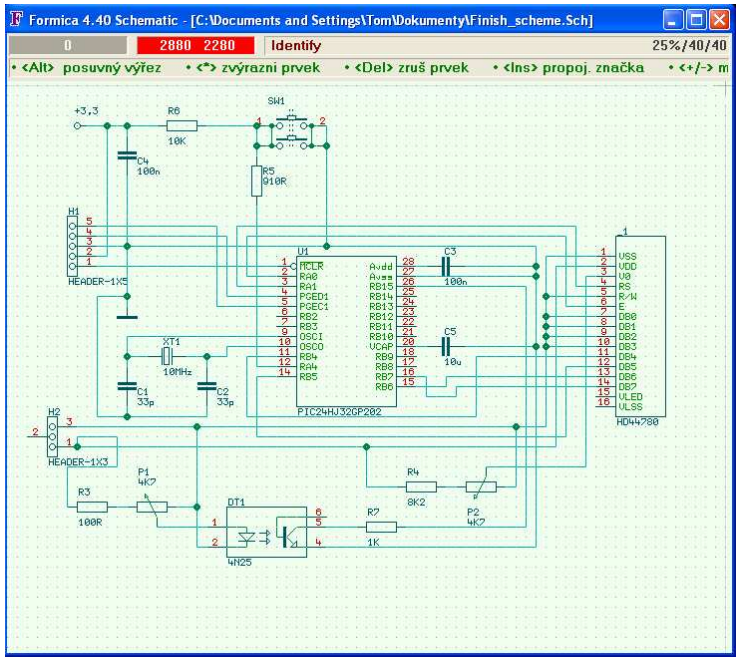
$$B = \frac{num1}{den} = \frac{n \cdot sum1 - sum2 \cdot sum3}{n \cdot sum4 - (sum2)^2} =$$
$$= \frac{n\left(\sum_{i=0}^{n} y_i \ln x_i\right) - \left(\sum_{i=0}^{n} \ln x_i\right)\left(\sum_{i=0}^{n} y_i\right)}{n\sum_{i=0}^{n} (\ln x)^2 - \left(\sum_{i=0}^{n} \ln x_i\right)^2}, \tag{4.9}$$

$$A = \frac{num2}{den} = \frac{sum4 \cdot sum3 - sum2 \cdot sum1}{n \cdot sum4 - (sum2)^2} =$$
$$= \frac{\left(\sum_{i=0}^{n} (\ln x_i)^2\right)\left(\sum_{i=0}^{n} y_i\right) - \left(\sum_{i=0}^{n} \ln x_i\right)\left(\sum_{i=0}^{n} y_i \ln x_i\right)}{n\sum_{i=0}^{n} (\ln x)^2 - \left(\sum_{i=0}^{n} \ln x_i\right)^2}. \tag{4.10}$$

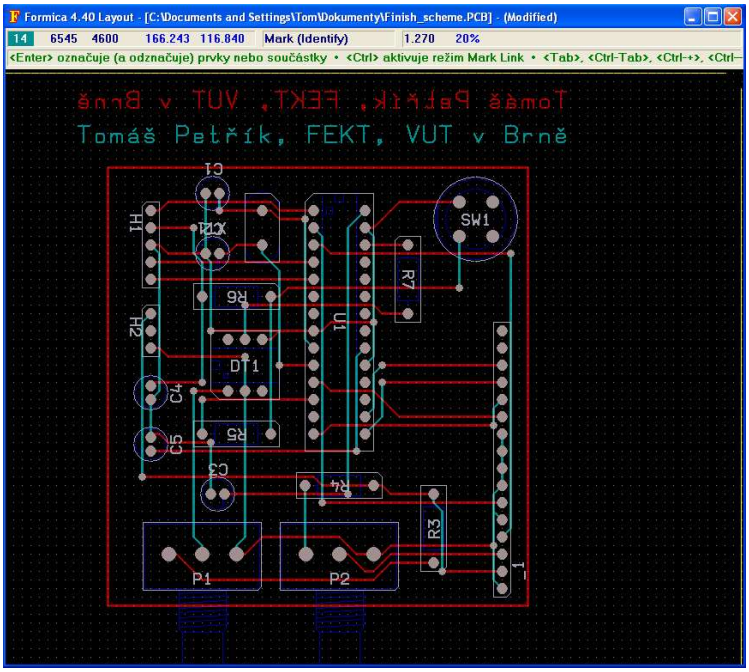
Funkce se počítá pro *n=5*, tedy pro počátečních 5 uzlových bodů odpovídajících úhlům natočení jezdce potenciometru pro α = 0°,30°,…,120°.

5 Návrh DPS pro obvod s mikroprocesorem

Návrh DPS pro měřicí obvod byl navržen v programu Formica. Nejdříve se nakreslil obvod v programu Formica Schematic (obr.10) a následně se vytvořila DPS (deska plošných spojů) v programu Formica Layout (obr.11). Návrh DPS je oboustranný a výstup návrhu je v příloze B. Strana A je na obrázku v příloze červeně. Strana B je modře a je stranou, která je osazena součástkami. Návrh DPS je vytvořen pro standardní velikosti pouzder a součástek.



Obr. 12 Návrh zapojení v programu Formica Schematic



Obr. 13 Návrh DPS v programu Formica Layout

6 Naměřené a vypočtené hodnoty

Logaritmický potenciometr s velikostí odporu 4,7kΩ se první proměřil digitálním multimetrem, poté se proměřil v zapojení dle schématu (obr.7) samotným mikroprocesorem PIC. Naměřené hodnoty napětí jsou v tabulkách (tab.1, tab.2) spolu s vypočítanými koeficienty *A, B* z rovnic (4.9) a (4.10).

Měření bylo provedeno celkem 6krát, tři měření voltmetrem a tři měření mikroprocesorem. Stupnice potenciometru byla rozdělena v rozsahu 0-270° po krocích o velikosti 30°, tedy celkem 9 uzlových bodů, které se proměřovaly. Po proměření prvních 5-ti bodů stupnice se vypočítala aproximační funkce, pomocí které se vypočítali následující předpokládané hodnoty zbylých 4 uzlových bodů. Čím vyšší je n, tedy počet uzlových bodů, z kterých se pak počítá náhradní funkce, tím je vyšší pravděpodobnost, že se náhradní funkce bude blížit té skutečné.

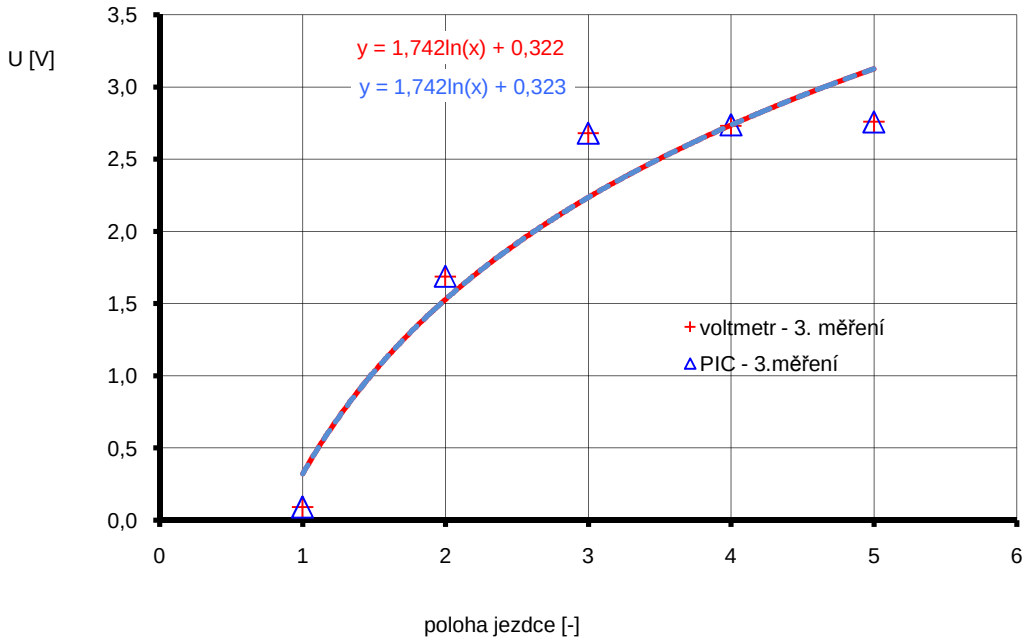
Charakteristiky dvou měření, jedno pro voltmetr a jedno pro PIC, byly vyneseny do společných grafů (obr.14 a obr.15). V prvním grafu (obr.14) je vyneseno prvních 5 naměřených uzlových bodů, pro které se následně vypočítala aproximační funkce, která je tam také vynesena spolu s tvary aproximačních funkcí pro hodnoty naměřené voltmetrem a mikroprocesorem. V druhém grafu (obr.15) je vyneseno 9 uzlových bodů, spolu s aproximačními funkcemi, které se počítaly ze všech naměřených uzlových bodů. Druhý graf je pro porovnání rozdílu výsledného tvaru aproximačních funkcí pro případ, kdy $n=5$ a $n=9$.

Tab. 1 Naměřené hodnoty voltmetrem

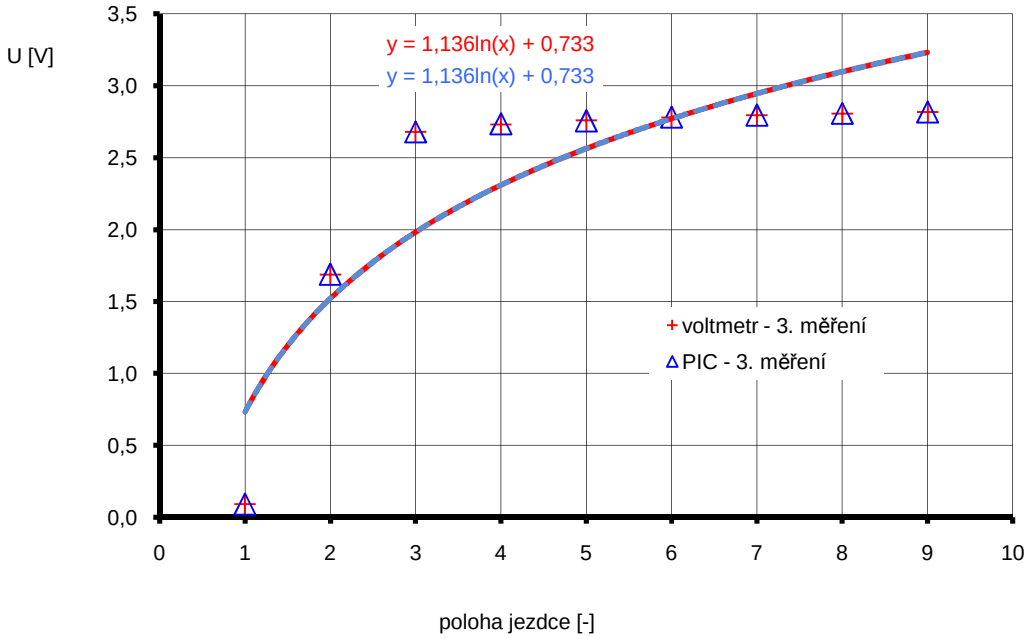
Poloha [-]	1	2	3	4	5	6	7	8	9	A	B
1.měření [V]	0,091	1,652	2,677	2,732	2,757	2,779	2,789	2,805	2,810	1,747	0,725
2.měření [V]	0,091	1,683	2,674	2,738	2,761	2,783	2,791	2,805	2,813	1,745	0,318
3.měření [V]	0,092	1,688	2,680	2,731	2,759	2,780	2,794	2,805	2,816	1,742	0,322

Tab. 2 Naměřené hodnoty mikroprocesorem PIC24HJ32GP202

Poloha [-]	1	2	3	4	5	6	7	8	9	A	B
1.měření [V]	0,090	1,661	2,684	2,736	2,759	2,783	2,796	2,807	2,813	1,742	0,312
1.měření [V]	0,090	1,627	2,681	2,736	2,764	2,780	2,791	2,805	2,817	1,755	0,299
3.měření [V]	0,092	1,689	2,679	2,736	2,756	2,782	2,796	2,808	2,817	1,742	0,323



Obr. 14 Závislost napětí na poloze jezdce U = f(α) pro prvních 5 uzlových bodů



Obr. 15 Závislost napětí na poloze jezdce U = f(α) pro všechny uzlové body

V následující tabulce jsou hodnoty z měření realizovaného mikroprocesorem PIC, reálné hodnoty napětí (R), teoretické hodnoty napětí (T), diference těchto dvou hodnot (d = R - T) a vypočítané koeficienty *A* a *B*.

Mikroprocesor PIC24HJ32GP202:

Tab. 3 Naměřené hodnoty mikroprocesorem PIC - kompletní měření

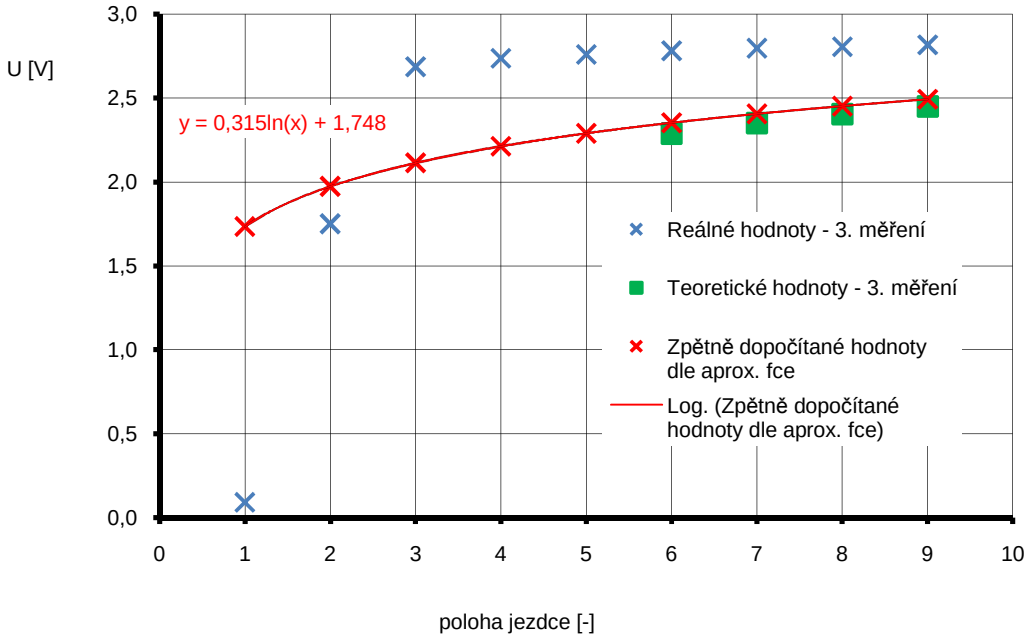
Měření	Napětí [V]	Kroky měření									Koeficienty	
		1	2	3	4	5	6	7	8	9	A	B
1	R	0,093	1,670	2,677	2,738	2,765	2,784	2,790	2,797	2,817	1,748	0,315
	T	-	-	-	-	-	2,255	2,312	2,361	2,403		
	d	-	-	-	-	-	0,530	0,470	0,437	0,415		
2	R	0,093	1,635	2,680	2,738	2,764	2,786	2,794	2,803	2,821	1,754	0,303
	T	-	-	-	-	-	2,241	2,296	2,343	2,383		
	d	-	-	-	-	-	0,545	0,498	0,460	0,437		
3	R	0,092	1,750	2,686	2,736	2,759	2,781	2,796	2,805	2,816	1,734	0,345
	T	-	-	-	-	-	2,288	2,351	2,404	2,450		
	d	-	-	-	-	-	0,493	0,445	0,400	0,365		

V tabulce níže (tab.4) jsou hodnoty vypočítané pro aproximační funkci, která má koeficienty *A* = 1,734 a *B* = 0,345 (3. měření z tab.3).

Zpětně dopočítané hodnoty s koeficienty ze 3. měření:

Tab. 4 Hodnoty vypočítané dle aprox. fce získané z mikroprocesoru PIC

Kroky	1	2	3	4	5	6	7	8	9
Napětí [V]	1,734	1,973	2,113	2,212	2,289	2,352	2,405	2,451	2,492



Obr. 16 Srovnání naměřených a teoretických hodnot napětí

Závěr

V bakalářské práci byla analyzována architektura mikrokontrolérů a dále byly probrány a shrnuty základní typy aproximačních funkcí. Na základě rozboru aproximací z hlediska složitosti implementace metody, velikosti chyby při aproximaci funkce metodou a náročnosti a rychlosti výpočtu mikroprocesorem, se ze zúženého výběru dvou typů metod nejmenších čtverců, lineární regrese a regrese polynomem, nakonec zvolila jako nejvhodnější metoda právě lineární regrese. Pro aplikaci této metody v případě výpočtu logaritmické závislosti, bylo nutno upravit vstupní hodnoty a to pouze přepočtem hodnot nezávisle proměnné na hodnoty jejich přirozených logaritmů. Složitější vzorce byly pro implementaci rozděleny na několik dílčích jednodušších vzorců, čímž se zpřehlednila samotná funkce pro výpočet aproximační funkce.

Zapojení mikroprocesoru pro měření bylo realizováno pomocí nepájivého pole a pro realizaci formou DPS, byl vytvořen oboustranný návrh DPS v programech Formica Schematic a Formica Layout, který je přiložen v příloze B.

Rychlost výpočtu aproximační funkce mikroprocesorem je dostačující, jelikož použitý mikroprocesor PIC patří mezi vyšší řady, které disponují především větší kapacitou pamětí (jak pro program, tak i pro data) a vyšším výpočetním výkonem.

Pro odečítaní napětí ze vstupu mikrontroléru, na který bylo přivedeno výstupní napětí z optočlenu, jenž galvanicky odděluje obvod mikroprocesoru s obvodem logaritmického potenciometru, byl použit 12-bitový A/D převodník. Tím pádem bylo možno odečítat hodnoty s přesností jednotek μV. Převodní charakteristika optočlenu však není lineární a tak je potřeba nastavení primárního obvodu prostřednictvím zvolených hodnot předřadných odporů (a v případě možnosti i nastavením vstupního napětí primárního obvodu) tak, aby byl pracovní bod optočlenu v přibližně lineární části převodní charakteristiky.

Po odečtení prvních pěti hodnot napětí při úhlech polohy jezdce 0°-120° byla vypočítána náhradní funkce, lineární regresí, kterou se pak předem dopočítali předpokládané hodnoty pro následující úhly v rozsahu 150° - 270°. Na displeji se pak tedy zobrazovaly hodnoty skutečné (naměřené), teoretické (dopočítané) a diference těchto dvou hodnot. Po dokončení měření se na displeji zobrazily hodnoty vypočítaných koeficientů A, B z rovnice náhradní funkce, tedy logaritmické závislosti. Výstup těchto měření je vidět v grafech, které jsou součástí této práce. Diference mezi hodnotami byly při úhlu 150° okolo 0,5 V, což je vcelku výrazná odchylka. Avšak v následujících krocích se diference neustále snižovala, tím pádem se naopak přesnost náhradní funkce zvyšovala. Užitečnost aplikace této práce v praxi by především záležela na požadované přesnosti odhadu průběhu závislosti měřeného prvku či systému.

Pokud by byla možnost nějakým způsobem navázat a pokračovat v této práci, pak bych navrhl rozšíření zařízení o další vstupní periferii a to např. numerickou klávesnici, pomocí které by bylo možno dalšími způsoby ovlivňovat nastavení, průběh měření a tím pádem i samotné výsledky měření. Pomocí numerické klávesnice by byla možnost volit mezi více aproximačními funkcemi, kterými se má reálná měřená závislost dopočítat. Doporučoval bych například implementaci regrese

polynomem, tedy další metodou nejmenších čtverců. Při výběru této metody by bylo možno prostřednictvím numerické klávesnice také zadávat požadovaný stupeň polynomu. V případě takovéto podoby měřicího zapojení by se mohlo dosáhnout zvýšení přesnosti výpočtu aproximační funkce, čímž by se snížila diference mezi skutečnými a teoretickými hodnotami.

Seznam použité literatury

[1] CNY17F. *Vishay Semiconductors* [online]. 2004 [cit. 2009-05-10].

[2] DEM 16217 SYH-LY : DataSheet, LCD Module. *Display Elektronik* [online]. 2003 [cit. 2009-04-25]. Dostupný z WWW: <http://www.display-elektronik.de/DEM16217SYH-LY.PDF>.

[3] DI JASIO , Lucio. *Programming 16-Bit PIC Microcontrollers in C : Learning to Fly the PIC 24*. [s.l.] : [s.n.], 2007. 379 s.

[4] FAJMON, Břetislav , RůŽIČKOVÁ, Irena. Matematika 3. [s.l.] : [s.n.], 2006. 254 s.

[5] GARDNER, Nigel. *PIC Micro MCU C : An introduction to programming the Microchip PIC in CSS C*. [s.l.] : [s.n.], 2002. 135 s.

[6] HRBÁČEK, Jiří. *Komunikace mikrokontroléru s okolím 1. díl.* Lektor: RNDr. Jiří Poš. 1999. vyd. Praha : BEN, 1999. 160 s. ISBN 80-86056-36-8.

[7] HRBÁČEK, Jiří. *Moderní učebnice programování PIC 2.díl*. [s.l.] : [s.n.], 2007. 140 s.

[8] PIC16F610/16HV610 PIC16F616/16HV616 Data Sheet : 14-Pin, Flash-Based 8-Bit CMOS Microcontrollers. Microchip [online]. 2008 [cit. 2008-11-25]. Dostupný z WWW: <www.microchip.com>.

[9] PIC24HJ32GP202/204 and PIC24HJ16GP304 Data Sheet : High-Performance, 16-bit Microcontrollers. *Microchip* [online]. 2007 [cit. 2009-04-20]. Dostupný z WWW: <www.microchip.com>

[10] *PICkit™ 2 : Microcontroller Programmer*. [s.l.] : [s.n.], 2007. 54 s.

[11] Wikipedie : Otevřená encyklopedie [online]. 2008 , 18.9.2008 [cit. 2008-11-25]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Metoda_nejmen%C5%A1%C3%ADch_%C4%8Dtverc%C5%AF>.

Seznam důležitých zkratek, značek a symbolů

Zkratky

ADC	Analog-to-Digital Converter
ALU	Arithmetic Logic Unit
DPS	Deska Plošných Spojů
FSR	File Select Register
GPR	General Purpose Registers
PC	Program Counter
PCH	Program Counter High
PCL	Program Counter Low
PCLATH	záchytný registr, pomocí něhož se mění hodnota registru PCH
PIC	Peripheral Interface Controllers
RISC	Reduced Instruction Set Computer
RWM	Read-Write Memory
SFR	Special Function Registers
WDT	Watchdog Timer

Seznam příloh

Příloha A: Zdrojový kód programu pro obsluhu mikroprocesoru

Příloha B: Návrh DPS

Příloha A

```
#include <p24hj32gp202.h>     // základní knihovna pro práci s PIC
#include <my_delay.h>         // definice prodlev
#include <math.h>             // knihovna matematických funkcí
#include <stdio.h>            // knihovna pro standardní vstupy/výstupy

_FOSCSEL(FNOSC_PRI & IESO_ON )
_FOSC(FCKSM_CSECMD & IOL1WAY_OFF & OSCIOFNC_ON & POSCMD_XT )
_FWDT( FWDTEN_OFF & WINDIS_OFF)
_FICD(ICS_PGD1)               // základní konfigurace mikrokontroléru

void lcd_init(void);          // předdefinování funkcí volaných ve funkci „main“
void enable_switch(void);
void lcd_write(char*,char);
void lcd_char(char);
void lcd_line(char);
void lcd_ready(char);
void initADC(int amask);
int readADC(int);
void mnc(void);

#define       E       LATAbits.LATA0      // definice názvů vstupů/výstupu, konstant
#define       RS      LATAbits.LATA1
#define       button  PORTAbits.RA4
#define       VREF    3.3

const unsigned char lcd_init_string[9] =
{0x20,0x20,0x80,0x00,0xc0,0x00,0x10,0x00,0x60}; // řetězec sloužící k inicializaci displeje

char          line_1[16],line_2[16];
int           amask,ch,result,step = 0;
unsigned char switch_count = 0;
float         x[10],y[10],y1[10],A,B;

 int main(void)
{
        amask = 0xfdff;
        ch = 9;                 // nastavení kanálu AN9 pro AD převod, (AN9 odpovídá
                                // RB15)
        AD1PCFGL = 0xffff;      // nastavení všech I/O pinů jako digitalní
        TRISA = 0x10;           // RA4 pin jako nastaven jako vstup (tlačítko „button“)
        TRISB = 0;              // nastavení všech pinů brány PORTB jako výstup
        E = 0;
        DELAY_MS(30);
        lcd_init();             // volaní funkce pro inicializaci displeje
        lcd_write("READY TO MEASURE",0);
        lcd_write("YOU MAY START...",1);
```

```
while(1)
{
    if ( button == 0 ) //pokud je stisknuto tlačítko pak se provede následující
        {
        if (switch_count < 8)    // opatření proti případným zákmitům, při stisku
                                 // tlačítka
          {
           switch_count ++;
           if (switch_count == 8)
                {
                if(step < 9)
                        {
                        if(step <= 4)
                                {
                                initADC(amask);
                                // volání funkce pro inicializaci AD převodníku
                                result = readADC(ch);
                                // volání funkce pro ADC
                                y[step] = VREF/4096*result;
                                // vyjádření výsledku ADC jako napětí  ve [V]
                                x[step] = step+1;
                                sprintf(line_1,"R:%05.3f T: NA",y[step]);
                                // funkce pro převod datového typu float na string
                                lcd_write(line_1,0);
                                lcd_write("d: NA ",1);
                                }
                        else      // po výpočtu MNČ
                                {
                                initADC(amask);
                                result = readADC(ch);
                                y[step] = VREF/4096*result;
                                sprintf(line_1,"R:%05.3f T:%05.3f", y[step],y1[step]);
                                sprintf(line_2,"d:%05.3f",(y[step]- y1[step]));
                                lcd_write(line_1,0);
                                lcd_write(line_2,1);
                                }
                        if(step==4)
                                mnc();  // volání funkce pro výpočet MNČ
                        }
                step++;
                if(step > 9) // po dokončení měření se zobrazí koeficienty A, B
                             // aproximační funkce
                        {
                        sprintf(line_1,"A: %07.5f",A);
                        sprintf(line_2,"B: %07.5f",B);
                        lcd_write(line_1,0);
                        lcd_write(line_2,1);
                        }
          }
        }
    }
    else
    {
       switch_count = 0;
    }
        }
return 0;
```

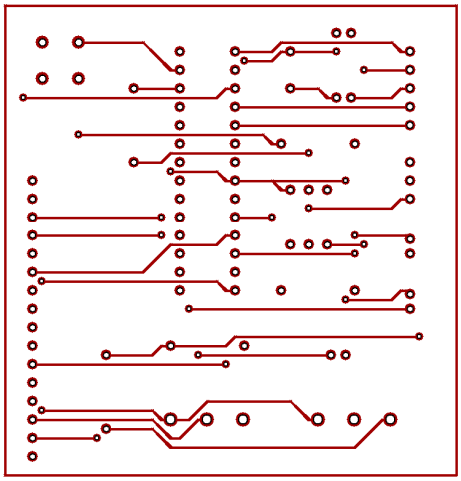
```
}

lcd_init()                              // inicializace LCD
{
        unsigned char i;

        for(i=0;i<=8;i++)
        {
                LATB = lcd_init_string[i];
                DELAY_CYCLES(1);
                enable_switch();
        }
}

enable_switch()                         // hodinový signál pro LCD
{
        E=1;
        DELAY_MS(5);
        E=0;
        DELAY_MS(5);
}


lcd_write(char *senpoint, char line)
{
        TRISB = 0;
        lcd_line(line);
        while(*senpoint != '\0')
        {
                lcd_char(*senpoint);
                senpoint++;
        }

}

void lcd_char(char letter)              // funkce posílající vždy jeden znak do řadiče
                                        // displeje, jenž se má zobrazit
{
        LATB = letter;                  // výpis symbolu na výstup PORTB
        DELAY_MS(5);
        RS = 1;                         // do řadiče LCD se posílá znak, jenž se má
                                        // zobrazit
        enable_switch();                // překlopení hodinového signálu
        LATB = LATB << 4;               // bitový posuv o 4 bity doleva, tj. odešle se nižší
                                        // část byte
        DELAY_MS(5);
        RS = 1;                         // RS v log. 1
        enable_switch();                // překlopení hodinového signálu pro řadič displeje

        lcd_line(char line)             // funkce, pro
{
        if(line == 0)
                {
                lcd_ready(0x01);
                lcd_ready(0x00);
                }
          else
```

```
                {
                lcd_ready(0xc0);
                }
}
        lcd_ready(char order)
{
        LATB = order;
        DELAY_MS(5);
        RS = 0;                         // do řadiče LCD se posílá příkaz
        enable_switch();                // překlopení hodinového signálu pro řadič displeje
        LATB = LATB << 4;               // posuv o 4 bity doleva, tj. odešle se nižší část byte
        DELAY_MS(5);
        RS = 0;                         // RS v log. 0
        enable_switch();
}


void initADC( int amask)
{
        TRISB=0xffff;                           // nastavení všech pinů brány PORTB jako
                                                // vstup
        AD1PCFGL = amask;                       // nastavení analogového vstupu pro ADC
        AD1CON1 = 0x04E0;                       // nastavení 12-bitového převodu ADC
        AD1CSSL = 0;                            // vypnutí skenování
        AD1CON2 = 0;                            // AVss and AVdd jsou použity jako Vref+
                                                // a Vref-
        AD1CON3 = 0x1F02;                       // Tad = 2 x Tcy = 125ns >75ns
        AD1CON1bits.ADON = 1;                   // zapnutí ADC
} //initADC

int readADC( int ch)
{
        AD1CHS0= ch;                            // 1. výběr kanálu pro ADC
        AD1CON1bits.SAMP = 1;                   // 2. spustí se vzorkování
        while (!AD1CON1bits.DONE);              // 3. čekání na dokončení AD konverze
                return ADC1BUF0;                // 4. čtení výsledku AD konverze
} // readADC

void mnc(void)                  // funkce pro výpočet aproximační funkce
{       unsigned char  i=0,n=5;
        float           sum1=0,sum2=0,sum3=0,sum4=0,sum5=0,num1,num2,den;

        for(i=0;i<=4;i++)
                {
                // rozdělení složitého vzorce na dílčí výpočty
                sum1+=logf(x[i])*y[i];
                sum2+=logf(x[i]);
                sum3+=y[i];
                sum4+=pow(logf(x[i]),2);
                sum5+=x[i];
                }

        num1=n*sum1-sum2*sum3;
        den=n*sum4-pow(sum2,2);

        B=num1/den;  // výpočet koeficientu B aprox.funkce
```

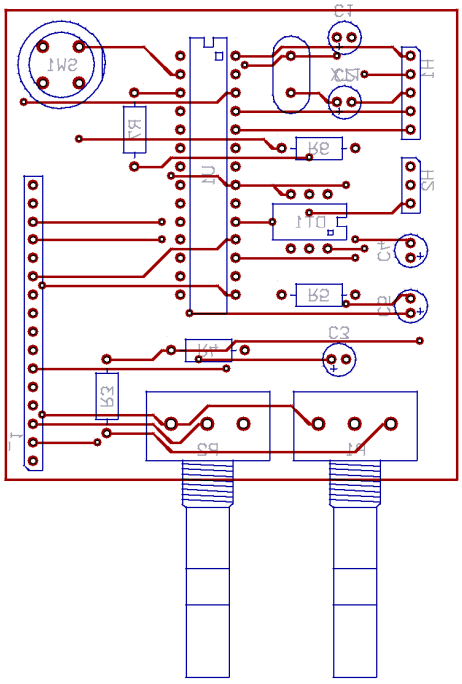
```
        num2=sum4*sum3-sum2*sum1;

        A= num2/den;  // výpočet koeficientu A aprox.funkce

        for(i=5;i<=9;i++)
                {
                        x[i]=i;
                        y1[i]=A+B*logf(x[i]); // výpočet následujících bodů charakteristiky
                                              // potenciometru, které se budou měřit
                }
}
```

Příloha B

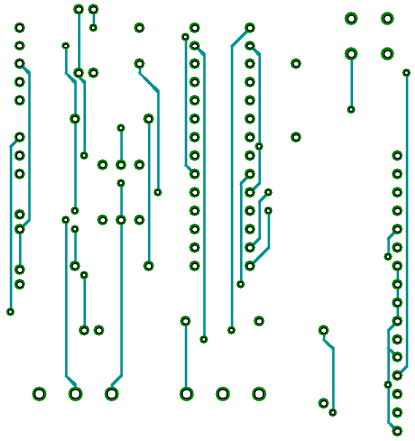
Tomáš Petřík, FEKT, VUT v Brně



Tomáš Petřík, FEKT, VUT v Brně



Tomáš Petřík, FEKT, VUT v Brně



Tomáš Petřík, FEKT, VUT v Brně