Středoškolská technika 2009

Setkání a prezentace prací

středoškolských studentů na ČVUT

Digitální spektrální analyzátor

Jakub Drs

SŘEDNÍ PRŮMYSLOVÁ ŠKOLA SDĚLOVACÍ TECHNIKY

Praha1, Panská 3

ANOTACE

Digitální spektrální analyzátor je měřící přístroj, který zpracovává vstupní analogový signál a vytváří z něho frekvenční spektrum. Využívá digitálního signálového procesoru k vypočtení spektra, které následně zobrazí na grafickém LCD displeji nebo je odešle po sériovém rozhraní do PC.

Analyzátor je schopen zpracovávat signál do 100kHz. Je možná volba pěti vzorkovacích kmitočtů v logaritmické řadě, podle nichž se vypočte vždy 128 spektrálních koeficientů. Dále jsou k dispozici dva typy kurzoru, frekvenční a napěťový pro měření přibližných hodnot napětí a frekvence. Celé zařízeni se ovládá pomocí dvou tlačítek a rotačního enkodéru. Na vstup je použit konektor typu BNC.

ANNOTATION

Digital spectral analyzer is a measuring instrument that transforms analog input signal into frequency spectrum. It uses digital signal processor for computing the spectrum, which will be displayed on graphical LCD or sent to PC, by the serial port.

The Analyzer is able to work with signal up to 100kHz. There are five options of sampling rate in logarithmic scale. Always 128 spectral coefficients are computed, depending on the sampling rate. In addition two types of cursor are available: frequency and voltage. Those are used for taking measures of frequency and voltage. The instrument is operated by two buttons and rotation encoder. There is a BNC connector on input.

1. OBSAH

Použitá literatura a zdroje informací

Microchip	33FJ32GP302	Datasheet
Philips	PCD8544	Datasheet
GM Electronic		katalog 2008
Skalický Petr	Aplikace signálových procesorů	ČVUT 2005

Použitý software

MPlab IDE v8.20

Překladač jazyka C pro dsPIC student edition v3.11

EAGLE 4.16r2

Obrázky

2. ÚVOD PRÁCE

Jádrem spektrálního analyzátoru je digitální signálový procesor dsPIC33FJ32GP302, který je dostatečně rychlý, je vybaven dostatkem paměti RAM a obsahuje rychlý 12bitový A/D převodník. Na vstupu zařízení je operační zesilovač v zapojení napěťového sledovače pro zvýšení impedance vstupu a oddělení od A/D převodníku (v případě přepětí na vstupu neodejde celý procesor). K zobrazení je využito grafického LCD displeje s řadičem PCD8544 ovládaného pomocí sériového rozhraní SPI. Tento displej má rozlišení 84x48bodů a používá se běžně ve starších mobilních telefonech NOKIA. K napájení slouží tři mikrotužkové baterie AAA.

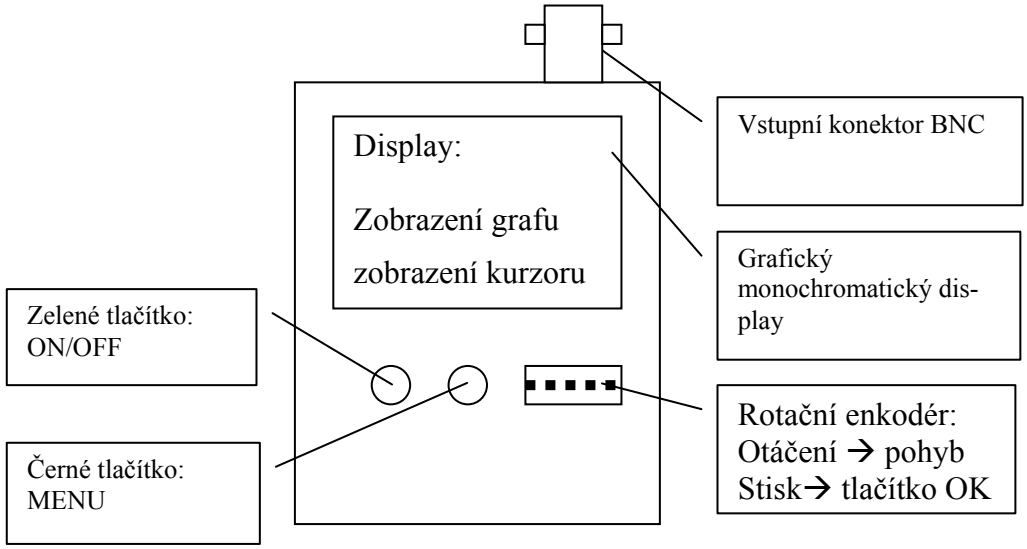
Technické parametry

Rozměry	63x28x117mm
Napájení	Tři tužkové baterie AAA
Maximální amplituda vstupního signálu	2Všš
Maximální frekvence vstupního signálu	100kHz

3. OVLÁDÁNÍ ZAŘÍZENÍ

Pohyb v menu

K zapnutí a vypnutí slouží zelené tlačítko, je to vypínač, který vše odpojí od baterie, takže nikam netečou žádné nanoampéry. Stiskem černého tlačítka se dostaneme do menu, ve kterém se pohybujeme pomocí kolečka. Jeho stisknutím zvolíme vybranou položku. Opětovným stiskem černého tlačítka se dostaneme o úroveň výš nebo zpět do zobrazovacího režimu.



1. Obrázek

Volby MENU

V MENU lze nastavit tyto položky:

Menu		
Rozlišení	Vertikální	1:1 , 2:1, 5:1, 10:1
	Horizontální	1:1 , 2:1
Kurzor	Napěťový, frekvenční	
Vzorkování	200, 100, 50, 20, 10 ksmp/sec	
Maximální amplituda	On/off	
PC	On/off	

Vertikální rozlišení

Nastavení amplitudy zobrazovaného signálu.

Horizontální rozlišení

- zobrazení 2:1

  Na displeji je zobrazeno celé spektrum. Kolečkem se pohybuje kurzorem.

- zobrazení 1:1

  Na displeji je podrobně zobrazena jen část spektra. Stiskem kolečka se přepíná mezi posouváním kurzorem a celým grafem spektra.

Kurzor

Přepíná mezi napěťovým a frekvenčním kurzorem.

Vzorkování

Nastavuje vzorkovací rychlost A/D převodníku.

Maximální amplituda

V tomto režimu je kurzor automaticky nastaven na spektrální složku s nejvyšší amplitudou.

PC

V režimu PC se odesílají spočtené spektrální koeficienty po sériovém portu. Zobrazení funguje normálně, avšak obnovování grafu je daleko pomalejší.

4. POPIS ZAJÍMAVÝCH SOUČÁSÍ

Procesor 33FJ32GP302

Jde o nový signálový procesor od firmy Microchip v dip pouzdře. Oproti námi ve škole používaným procesorům 30F má navíc několik užitečných funkcí. Instrukční sada a DSP jádro zůstávají téměř stejné. Procesory 33F jsou však o něco rychlejší (40MIPS). Především pracují na jiné logické úrovni a to 3,3V, což přináší pár problémů s programováním, avšak pro převod úrovně postačí převodník z 3,6V zenerovy diody a 50Ω rezistoru. Dále je nutné dát kondenzátor na vnitřní stabilizátor pro jádro mezi piny VSS a VDDCORE a to i při programování.

Asi nejužitečnější nová funkce je DMA (direct memory access) ta umožňuje přenos dat mezi pamětí RAM určené pro DMA a SFR registry mimo jádro procesoru. To znamená, že můžeme vzít např. blok dat zadat DMA aby ho odeslalo po UARTu a mezitím zpracovávat zcela jiný program. Až budou odeslány všechny znaky DMA vyvolá přerušení. Je samozřejmě možný i opačný proces. Zadat DMA aby přijalo po UARTu blok dat, zapsalo ho do paměti a pak vyvolalo přerušení. Je dokonce možné použít tzv. ping pong buffer. Znamená to že buffery jsou dva a střídavě se zapisuje nebo čte do jednoho nebo do druhého. Je to dobré k tomu, že zatímco se jeden buffer např. přijímá po UARTu pomocí DMA, můžeme druhý buffer zároveň zpracovávat. To většinou trvá kratší dobu než jeho přijímání a proto ihned po přijetí celého bloku se oba buffery prohodí a zatímco se nově přijatý buffer zpracovává druhý se zároveň vyplňuje daty. Já tuto funkci využívám pro čtení z vnitřního A/D převodníku a zatímco se jedny data vzorkují, druhé již zpracovává algoritmus FFT. Díky tomu může být vzorkování nepřerušované.

Mapování portů periferií

Je také velice užitečná funkce umožňující softwarově umístit vstupní a výstupní porty periferií téměř na kterékoliv piny procesoru, což je velmi užitečné při vývoji plošného spoje. Piny na které je možné periferie umístit mají označení RP0 – RP31. Umístění se provádí zápisem do registrů RPINR a RPOR. Liší se pro vstupy a výstupy. Každý vstup má své místo v nějakém RPINR registru kam se zapíše 5 bitové číslo udávající číslo RP. Pro na-

stavení vstupu periferie tedy do patřičného RPINR, daného názvem vstupu.

```
RPINR18bits.U1RXR=0;                //vsup UARTu je na RP0
```

Naopak každý RP pin má své místo v patřičném RPOR registru a každý výstup má své číslo. Pro nastavení výstupu tedy toto číslo zapíšeme do patřičného RPOR registru.

```
RPOR0bits.RP1R=3; //U1TX má číslo 3 a bude ne RP1
```

Toto nastavení má několik možností zabezpečení proti náhodnému chybnému přepsání. Podobně jako zápis do EEPROM. V programovacím jazyce C pro dsPIC jsou však defaultně vypnuté.

A/D převodník

Převodník v tomto procesoru může pracovat ve dvou režimech. Může být buď 12bitový o maximální rychlosti 500ksmp/sec nebo 10bitový o maximální rychlosti 1,1Msmp/sec.

LCD

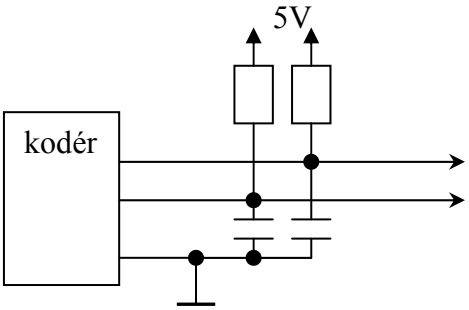
Jde o display vymontovaný ze staré nokie 3310 stejný je i v mnoha dalších typech. Tento display je monochromatický s rozlišením 48x84 bodů. Důležité je, že má již integrovaný řadič PCD8544, od kterého je ke stažení datasheet. Ovládá se sériovým rozhraním SPI a třemi piny. Po zapnutí napájení je potřeba provést v krátkém čase reset LCD. Poté se musí nastavit řadič. Byte odeslaný po SPI může znamenat buď instrukci pro řadič nebo data podle nastavení pinu DC. Nejmenším prvkem, který je možný najednou zobrazit je jeden sloupec 1x8 bodů, neboli jeden byte odeslaný po SPI. Každý bit znamená jeden pixel. Každý tento sloupec má svou vertikální a horizontální adresu. Na display se tedy vejde 6x84 těchto sloupců. Toto LCD je navrženo pro zobrazování textu, ale i grafiky proto je vybaveno vertikálním a horizontálním adresováním. Znamená to, že po odeslání jednoho bytu dat se adresa automaticky posune buď o jeden sloupec doprava, což je vhodné pro zobrazování textu, nebo o jeden sloupec dolu, což je vhodné pro zobrazování grafiky. Podrobnější informace lze nalézt v datasheetu. Pro úplnost je nutné dát kondenzátor 1μF mezi piny VOUT a GND. V mém schematu tento kondenzátor chybí protože jsem ho připojil přímo na LCD.

Rotační enkodér

Rotační enkodér je takové to věčné kolečko, kterým můžeme otáčet stále dokola. Známe ho například z kolečka u myši nebo nových osciloskopů. Mívá v sobě často zabudované i obyčejné tlačítko. Funguje tak, že při otáčení generuje dva obdélníkové signály které jsou vůči sobě posunuty o čtvrt periody doprava nebo doleva podle směru otáčení.

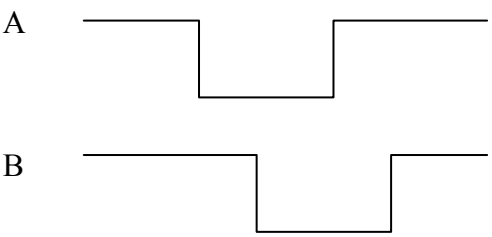
Fyzicky má obvod tři vývody. Z nichž jeden vývod je společný a zbylé dva jsou proti němu spínány. Na tyto vývody je třeba dát kondenzátory proti zákmitům přibližně 22nF.

Obvyklé zapojení:



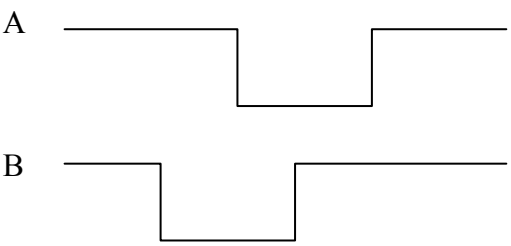
2. obrázek

Otáčka doprava



3. obrázek

Otáčka doleva



4. obrázek

5. PROGRAMOVÉ VYBAVENÍ

Programovací jazyk C pro dsPIC

Program byl vytvořen v integrovaném vývojovém prostředí MPlab IDE v programovacím jazyce C pro dsPIC studentské verzi. Toto prostředí vyvíjené firmou Microchip je volně šiřitelné stejně jako i překladač jazyka C. Práce v tomto jazyce je daleko snazší než v assembleru a kód je o mnoho přehlednější. Krom toho již tento jazyk obsahuje běžné matematické operace jako sin, cos a odmocniny, ale i mnoho knihoven pro práci s DSP jádrem. Jsou to např. filtry FIR, rutiny pro práci s vektorovými veličinami, nebo již zmíněný algoritmus FFT. Díky softwarovému zásobníku je také relativně snadná kombinace jazyka C s assemblerem pokud je potřeba napsat rutinu pracující s DSP jádrem nebo část programu kterou v jazyce C nelze snadno vytvořit.

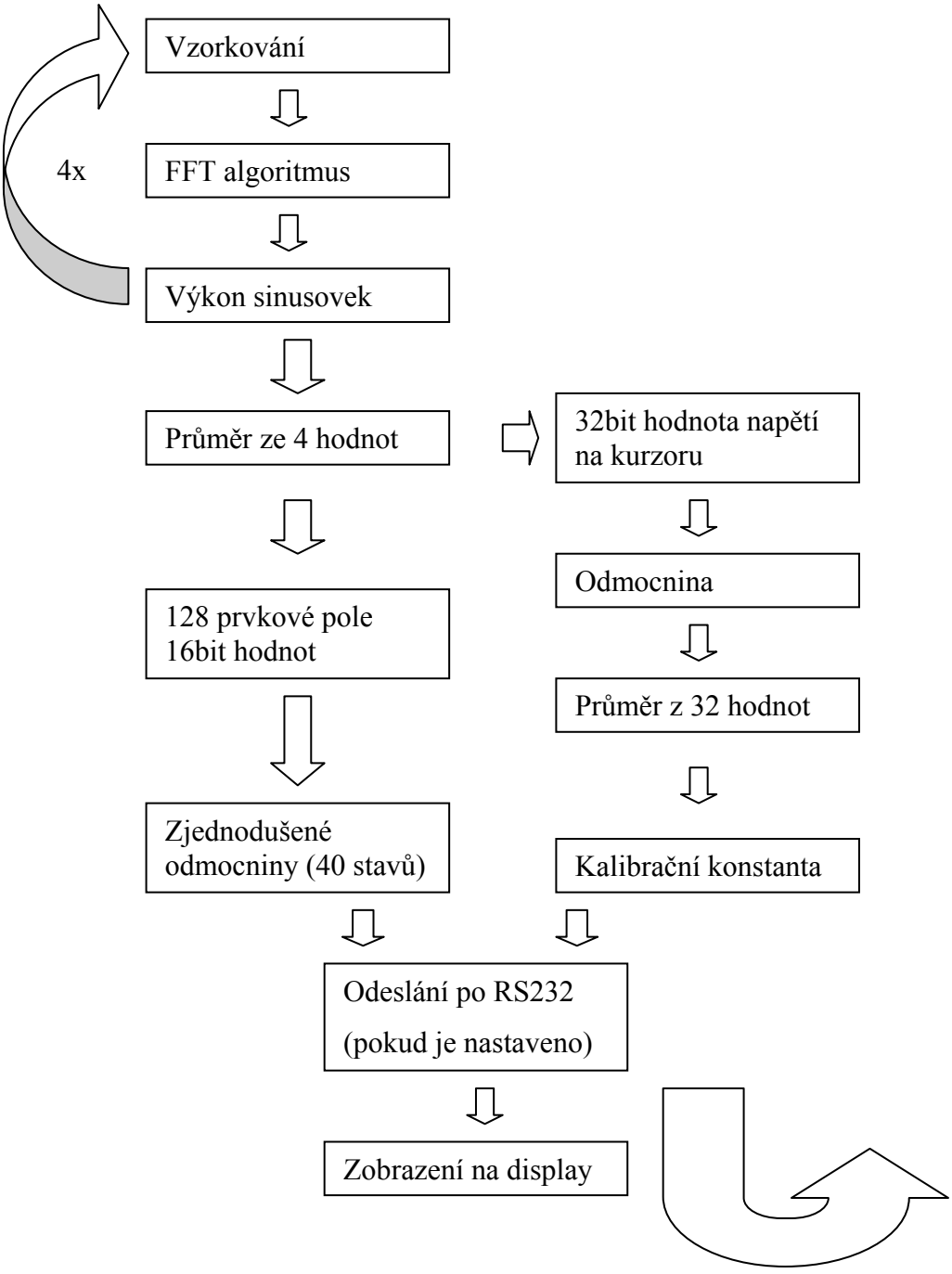
Stručný popis programu

Zpočátku nejprve proběhne čtyřikrát za sebou vzorkování, výpočet FFT a výkonů sinusovek. Vzorkování probíhá o něco déle než samotný výpočet, takže ve zbývajícím čase se zjišťuje, má li se jít do menu. Vytváří se průměr ze čtyř hodnot, aby signál nebyl zašuměný. Liší se zpracování hodnot výkonů sinusovek pro zobrazení na LCD v podobě čáreček(pole 128 hodnot) a pro hodnotu kurzoru zobrazenou číselně(32 bitová hodnota). Hodnotu na kurzoru je třeba uchovávat v plném rozlišení, jelikož po odmocnění zbude vždy jen polovička bitů. Hodnotu kurzoru je po odmocnění následně ještě nutno pořádně zprůměrovat, aby hodnota neustále neskákala. Zobrazí se vždy až průměr z 32 hodnot. Tyto hodnoty se následně odesílají na LCD případně i na sériový port. Zařízení proto pracuje v režimu PC zcela stejně, akorát o něco pomaleji.

Ovládací prvky jako jsou tlačítka a rotační enkodér jsou řízeny v přerušení. Vlastnímu programu jsou poté předávány informace pomocí řídící struktury různých příznakových bitů a hodnot.

Pokud se má vstoupit do menu přestává se vzorkovat a zobrazí se text menu. Text se překresluje vždy po pootočení enkodérem, nebo stisku tlačítka. Informace o zvolených položkách jsou opět ukládány do řídící struktury.

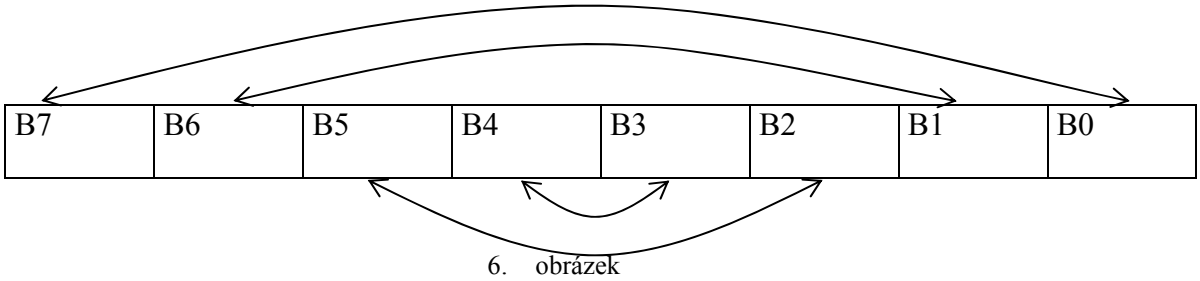
Zjednodušený vývojový diagram



5. obrázek

6. POPIS ALGORITMU FFT (FAST FOURIER TRANSFORMER)

Tento algoritmus slouží k převedení například navzorkovaného signálu na řadu spektrálních koeficientů. Pracuje s komplexním polem (pole je dvourozměrné – každá hodnota se skládá z reálné a imaginární části) s počtem prvků 2^n ,pro které je v programovacím jazyce C pro dsPIC již hotový datový typ `fractcomplex.` Toto pole je vstupem i výstupem algoritmu. Napřed se do něho uloží navzorkovaný signál ale jen do reálných složek, imaginární by měli zůstat nulové. Toto pole se pošle na vstup FFT algoritmu. (Tento algoritmus je obsažen v knihovnách, dodávaných firmou Microchip společně s Jazykem C pro dsPIC.) Poté, co pole projde algoritmem je třeba ještě proházet koeficienty podle bitově reverzní posloupnosti tj. první bit se prohodí s posledním, druhý s předposledním.



6. obrázek

Výsledkem toho je již pole které obsahuje reálné a imaginární složky jednotlivých spektrálních koeficientů. Pokud chceme k tomu získat ještě jejich absolutní hodnotu musíme reálnou a imaginární složku vektorově sečíst. Samozřejmě jde o starou dobrou Pythagorovu větu.

$$a^2 + b^2 = c^2$$

Její výpočet však není zcela snadný protože je třeba počítat s odmocninami.

$$c = \sqrt{a^2 + b^2}$$

Programovací jazyk C samozřejmě odmocniny podporuje avšak funkce se zpracovává poměrně dost dlouho. Ve většině případů však potřebujeme jen porovnat amplitudy koeficientů takže stačí počítat jen s hodnotou bez odmocniny, která odpovídá výkonu sinusovky dané koeficientem. Ovšem v mém případě spektrálního analyzátoru tomu tak samozřejmě nebylo. Musí se vypočítat všechny amplitudy, ale s poměrně nízkou přesností pouze 40 hodnot (na LCD graf zabírá 40 řádek). Proto si nejprve vypočítám pro těchto 40 hodnot

druhé mocniny, se kterými následně porovnávám výkony složek. Jediná hodnota kurzoru je vypočtena opravdovou odmocninou z 32bitového čísla.

7. HARDWARE

Zařízení je vyrobeno na jednostranné desce plošného spoje navržené v programu Eagle. Deska byla vyrobena ve školních laboratořích fotocestou. Je použita kombinace obyčejných a SMD součástek. Procesor a operační zesilovač jsou v dip pouzdrech umístěny v paticích, rezistory a kondenzátory jsou SMD pro ušetření místa na desce.

Napájení tvoří tři tužkové baterie tedy 4,5V. Prochází přes Schottkyho diodu, aby nemohlo dojít k zničení přístroje přepólováním a dále již do stabilizátoru 3,3V. Z něj se celé zařízení napájí a tvoří i referenční napětí A/D převodníku. Napájení pro A/D převodník je odděleno rezistorem 2Ω a filtrováno kondenzátorem 1000μF/6,3V.

Tlačítka, display, napájení a vstupní konektor jsou připojeny pomocí kolíkových lišt. Vše je umístěno v plastové krabičce, která má i dvířka pro držák na baterie, takže jdou baterie vyměnit bez použití šroubováku

8. PŘÍLOHY

Plošný spoj

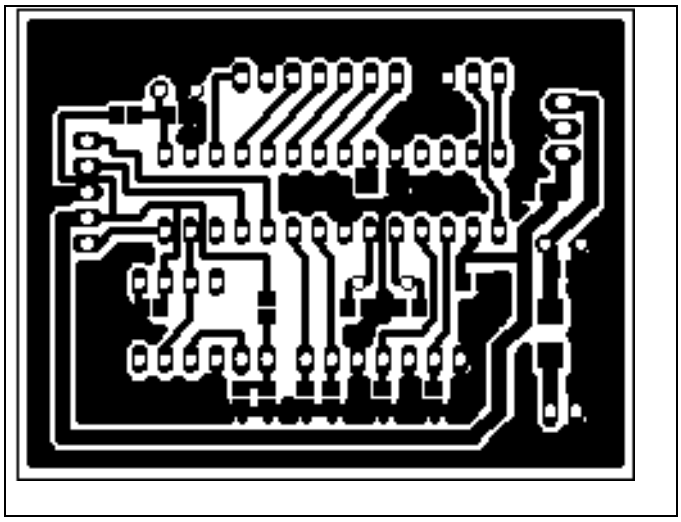
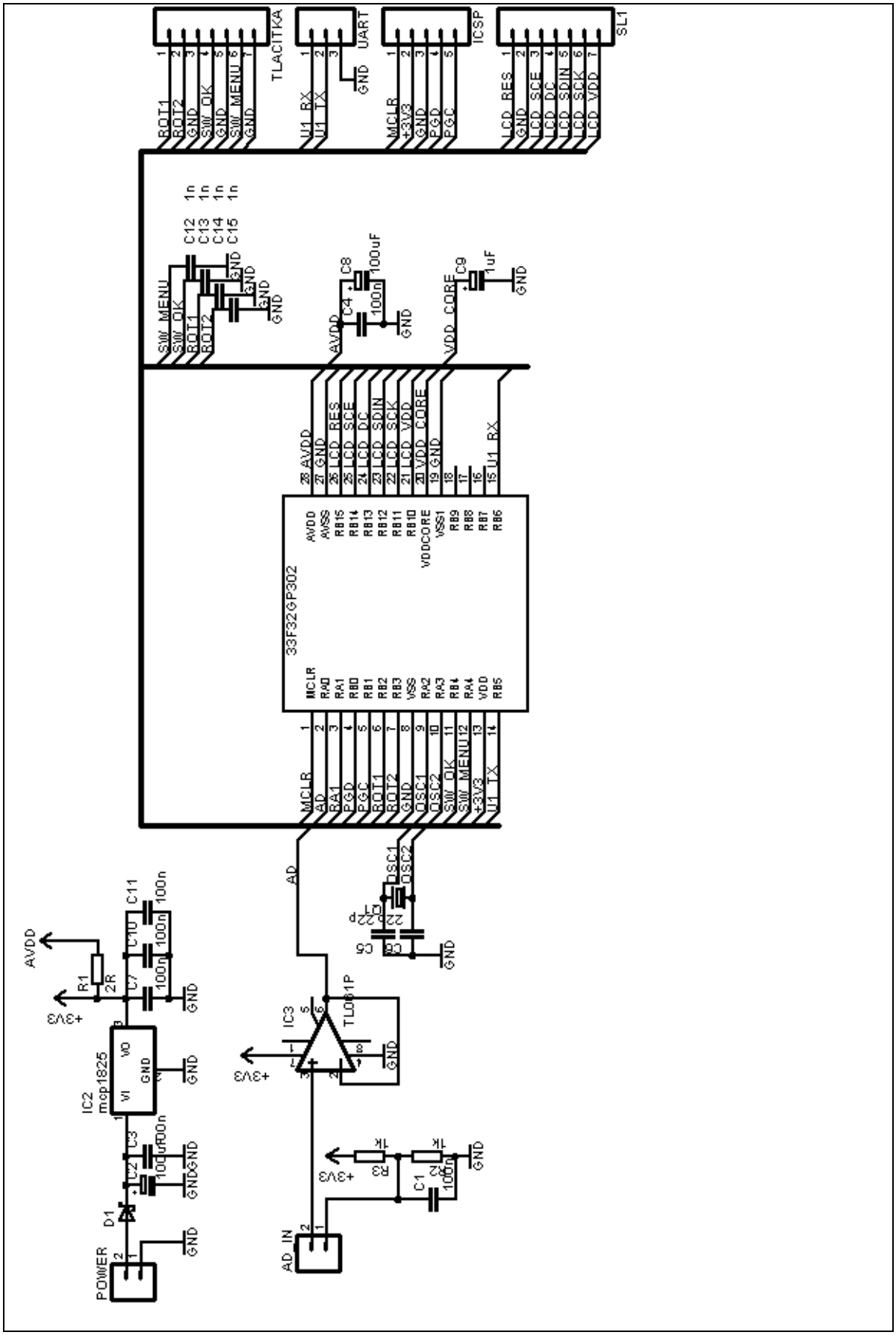


Schéma zařízení



Vybrané části programu

Main.c

```
#include "common.h"
#include "control.h"

//FUSES
_FOSCSEL(FNOSC_FRC);
_FOSC( FCKSM_CSECMD & IOL1WAY_OFF & OSCIOFNC_OFF & POSCMD_XT );
_FWDT(FWDTEN_OFF);
_FICD( BKBUG_OFF & COE_OFF & JTAGEN_OFF & ICS_PGD1 );


//EXTERNI FUNKCE
        extern void zobraz_LCD(void);
        extern void generate_sin(void);
        extern void generate_signal(void);
        extern char procesFFT(void);
        extern void InitSPI(void);
        extern void Init_LCD(void);
        extern void testlcd(void);
        extern void initDma0(void);
        extern void initAdc1(void);
        extern void Init_coder_int(void);
        extern void menu(void);
        extern void init_uart(void);
        extern void serial_send(void);


//MISTNI FUNKCE
        void menu(void);
        void ProcessADCSamples(int * AdcBuffer);
//EXTERNI PROMENE
        extern fractcomplex sigCmpx[FFT_BLOCK_LENGTH]
        __attribute__ ((space(ymemory),far,aligned(FFT_BLOCK_LENGTH * 2 *2)));



//********************MAIN*************************/

void main (void)
{
//                  INICIALIZACE OSCILATORU
//                  krystal f=4MHz
// Configure Oscillator to operate the device at 40Mhz
// Fosc= Fin*M/(N1*N2), Fcy=Fosc/2
// Fosc= 4M*40/(2*2)=80Mhz for 4M input clock
        PLLFBD=78;                                  // M=80
        CLKDIVbits.PLLPOST=0;                       // N1=2
        CLKDIVbits.PLLPRE=0;                        // N2=2
// clock switching to incorporate PLL
        __builtin_write_OSCCONH(0x03);              // Initiate Clock Switch to Primary
                                                    // Oscillator with PLL (NOSC=0b011)
        __builtin_write_OSCCONL(0x01);              // Start clock switching
        while (OSCCONbits.COSC != 0b011);           // Wait for Clock switch to occur
        while(OSCCONbits.LOCK!=1);                  // Wait for PLL to lock
```

```
//                  INICIALIZACE PORTU
        PMCON=0;        //Paralel port OFF
        CMCON=0;        //comparator OFF
        AD1PCFGL=0xFFFF;

//                  CONTROL DEFAULT
        Control.pohyb =0;
        Control.vzorkovani=2;
        Control.poloha=0;
        Control.flags.h_rozliseni=1;
        Control.v_rozliseni=0;
        Control.kurzor=64;
        Control.flags.v_f=1;
        Control.flags.PC=0;
        Control.flags.max_amp=0;
//*****************************/
        InitSPI();

        Init_LCD();

        testlcd();

        initAdc1();

        initDma0();

        init_uart();

        Init_coder_int();

        Control.pohyb =0;

        while(1)
        {

//      generate_signal();              //generator obdelniku vygeneruje signal a predhodi ho FFT

//      generate_sin();                 //generator sinusovky pro moznost simulace FFT

//*******************MENU***************/
if(Control.flags.menu)              //Ma se vstoupit do menu?
        {                                                       //pokud ano
        Control.flags.menu=0;   //vynulovani priznaku vstoupit do menu
        Control.flags.ad_stop=1; //stop ad prevodu
        menu();
        Control.flags.ad_stop=0;
        IFS0bits.DMA0IF = 1;    //skoc do preruseni
        }

if(Control.pohyb)                   //ma se pohnout kurzorem?
        {
        if(Control.flags.h_rozliseni) Control.flags.pohyb_kurzor=1;
        if(Control.flags.pohyb_kurzor)
                        {
                        Control.kurzor+=Control.pohyb;
                        if(Control.kurzor>150)    Control.kurzor=0;
                        if(Control.kurzor>127)    Control.kurzor=127;
                        }
                else
                        {
```

```
                        Control.poloha-=Control.pohyb;
                        if(Control.poloha>150)    Control.poloha=0;
                        if(Control.poloha>44)     Control.poloha=44;
                        }
        Control.pohyb=0;
        }
if(Control.flags.PC)                    //ma se odeslat na PC
        {
        if(Control.flags.ad_stop)
                serial_send();
        Control.flags.PC_odeslano=1;
        }
//******************ZOBRAZENI***********/
if(!(!Control.flags.PC_odeslano & Control.flags.PC))        //nejdrive se odesila na PC pak zobrazuje
        if(Control.flags.ad_stop)
                {
                zobraz_LCD();           //zobrazeni na LCD
                Control.flags.ad_stop=0;
                Control.flags.PC_odeslano=0;
                IFS0bits.DMA0IF = 1;    //skoc do preruseni
                }

        }
}


//Tato funkce je volana z preruseni od DMA
//Pripadne ji vola generator sinusovky
//cela tato funkce musi probehnout rychleji nez vzorkovani
void ProcessADCSamples(int * AdcBuffer)
        {

        unsigned int i;
//return;
        fractcomplex *p_cmpx= &sigCmpx[0];
                for ( i = 0; i < FFT_BLOCK_LENGTH; i++ ) //move data into sigCmpx
                {
                        (*p_cmpx).real = (*AdcBuffer++);
                        (*p_cmpx++).imag = 0x0000;
                }
        if(procesFFT())  //funkce 4x spocita FFT z SigCmpx
                                                //vysledek uklada do abs_prumer

                                                //po 4 probehnuti vraci 1
                Control.flags.ad_stop=1;

        }
```

Řídicí struktura

```
struct
{

signed char pohyb;                  //pootoceni koleckem kladna doprava zaporna doleva
unsigned char kurzor;               //aktualni poloha kurzoru
unsigned char poloha;               //aktualni poloha grafu
unsigned char v_rozliseni;          //nastaveni vertikalniho rozl.
unsigned char vzorkovani;           //nast. vzorkovac9 rychlosti

struct
        {
        unsigned ad_stop :1;        //zastavit ad prevod
        unsigned menu :1;           //stisknuto tlacitko menu
        unsigned OK :1;             //stisknuto tlacitko OK
        unsigned h_rozliseni :1;    //prepina vertikalni a horizontalni zobrazeni
        unsigned v_f :1;            //prepina napetovy a frekvencni kurzor
        unsigned pohyb_kurzor :1;   //prepina pohyb kurzorem a grafem
        unsigned max_amp :1;        //zapina max ampl.
        unsigned PC :1;             //zapina PC
        unsigned PC_odeslano :1;    //dokonceni odesilani

        }flags;
} Control;

//      DEFINITION
#define SW_OK  PORTBbits.RB4        //tlacitko OK
#define SW_MENU PORTAbits.RA4       //tlacitko menu
```

Menu.c

```
#include "common.h"


extern

#include "control.h"

extern int zobraz_menu(char *text,char radky,char poc);



void menu(void)

{

const char menu[6][14]={

{"   *MENU*   "},

{" rozliseni   "},

{" kurzor      "},

{" vzorkovani  "},

{" max.amplituda"},
```

```
{" PC"},
},
rozliseni[6][14]={
{" *ROZLISENI*  "},
{"vertikalni"},
{"horizontalni"},

},
kurzor[6][14]={
{"   *KURZOR*   "},
{"napeti"},
{"frekvence"},

},

vzorkovani[6][14]={
{" *VZORKOVANI* "},
{"200 ksmp/s"},
{"100 ksmp/s"},
{"50 ksmp/s"},
{"20 ksmp/s"},
{"10 ksmp/s"},
},
max_amplituda[6][14]={
{"MAX.AMLITUDA"},
{"off"},
{"on"},
},
h_rozliseni[6][14]={
{"HORIZONTALNI R"},
{"1:1"},
{"2:1"},
},

v_rozliseni[6][14]={
{"VERTIKALNI R. "},
{"1:1"},
{"2:1"},
{"5:1"},
{"10:1"},
```

```
},

PC[6][14]={
{"    *PC*    "},
{"off"},
{"on"},
};


int pozice;

delay_xms(100);
while(1)
        {
        pozice=zobraz_menu(menu,6,1);
        if(pozice==0) break;
        switch(pozice)
                {
                case 1:                                 //ROZLISENI
                        pozice=zobraz_menu(rozliseni,3,1);
                        if(pozice==0) break;
                        if(pozice==1)
                                {
                                pozice=zobraz_menu(v_rozliseni,5,Control.v_rozliseni+1);
                                if(pozice==0) break;
                                Control.v_rozliseni=pozice-1;
                                }
                        else
                                {
                                pozice=zobraz_menu(h_rozliseni,3,Control.flags.h_rozliseni+1);
                                if(pozice==0) break;
                                Control.flags.h_rozliseni=pozice-1;
                                }

                        break;
                case 2:                         //KURZOR
                        pozice=zobraz_menu(kurzor,3,Control.flags.v_f + 1);
                        if(pozice==0) break;
                        Control.flags.v_f = pozice-1;
                        break;
```

```
                case 3:                 //VZORKOVANI
                        pozice=zobraz_menu(vzorkovani,6,Control.vzorkovani);
                        if(pozice==0) break;
                        Control.vzorkovani=pozice;
                        switch(pozice)
                                {
                                case 1:
                                        PR3=200;
                                        break;
                                case 2:
                                        PR3=400;
                                        break;
                                case 3:
                                        PR3=800;
                                        break;
                                case 4:
                                        PR3=2000;
                                        break;
                                case 5:
                                        PR3=4000;
                                        break;
                                }
                        break;
                case 4:                 //MAXIMANLI AMPLITUDA
                        pozice=zobraz_menu(max_amplituda,3,Control.flags.max_amp+1);
                        if(pozice==0) break;
                        Control.flags.max_amp=pozice-1;
                        break;
                case 5:                 //PC CONNECTION
                        pozice=zobraz_menu(PC,3,Control.flags.PC+1);
                        if(pozice==0) break;
                        Control.flags.PC=pozice-1;
                        break;
                }

        }
        while(Control.flags.menu);
        Control.flags.menu=0;

}
```