Středoškolská technika 2012

Setkání a prezentace prací středoškolských studentů na ČVUT

KALKULÁTOR S LCD DISPLEJEM

Lukáš Herudek

Střední průmyslová škola elektrotechniky a informatiky, Ostrava, příspěvková organizace
Kratochvílova, 7/1490, Ostrava - Moravská Ostrava,702 00

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 10 - Elektrotechnika, elektronika a telekomunikace

Kalkulátor s LCD displejem

Calculator with LCD display

Autor: Lukáš Herudek

Škola: Střední průmyslová škola elektrotechniky a informatiky, Ostrava

Ostrava, 2012

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně, použil jsem pouze podklady (literaturu, SW atd.) uvedené v přiloženém seznamu a postup při zpracování a dalším nakládání s prací je v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Ostravě, dne 27.2.2012

X

Lukáš Herudek

Poděkování

Chtěl bych poděkovat panu Ing. Karlu Gogolkovi za podporu a paní Ing. Pavlíně Pavlové za navedení do práce SOČ.

ANOTACE

Výsledkem tohoto projektu je snadno ovladatelný kalkulátor, který je možné snadno reprodukovat. Můžete jej také kdykoli přeprogramovat a vytvořit tak i své vlastní funkce, se kterými můžete počítat. Použil jsem běžně vyskytující se součástky, takže není problém je koupit. Při stavbě tohoto projektu jsem si rozšířil znalosti v oblasti elektrotechniky, zvláště pak programování mikroprocesorů.

Klíčová slova: LCD 2x16; ATMEGA16; kalkulátor; mikroprocesor; jazyk C; maticová klávesnice.

ANNOTATION

The result of this project is simply-controlled calculator, which is easy to reproduce. You can also re-program it and you can create your own functions which you can count with. I used normally offered components so there is no problem to buy them. While making this project I extended my knowledge about electronics especially programming microprocessors.

Key words: LCD2x16; ATMEGA16; calculator; microprocessor; C language; matrix keyboard

Obsah

Seznam obrázků

1. ÚVOD

Cílem tohoto projektu bylo vytvořit levný školní kalkulátor s mnoha funkcemi. Koncept vychází z mého prvního kalkulátoru, který byl ovšem příliš velký, měl velkou spotřebu a pouze pár základních funkcí. Proto jsem se rozhodl vytvořit druhou verzi, lepší, praktičtější, pokročilejší.

Pro tvorbu softwaru jsem použil AVR Studio od firmy Atmel. Pro programování jsem zvolil jazyk C. K ovládání slouží dvě nezávislé maticové klávesnice, které jsem zvolil pro svou hardwarovou nenáročnost a tím i snížení ceny, samozřejmě na úkor náročnosti programu.

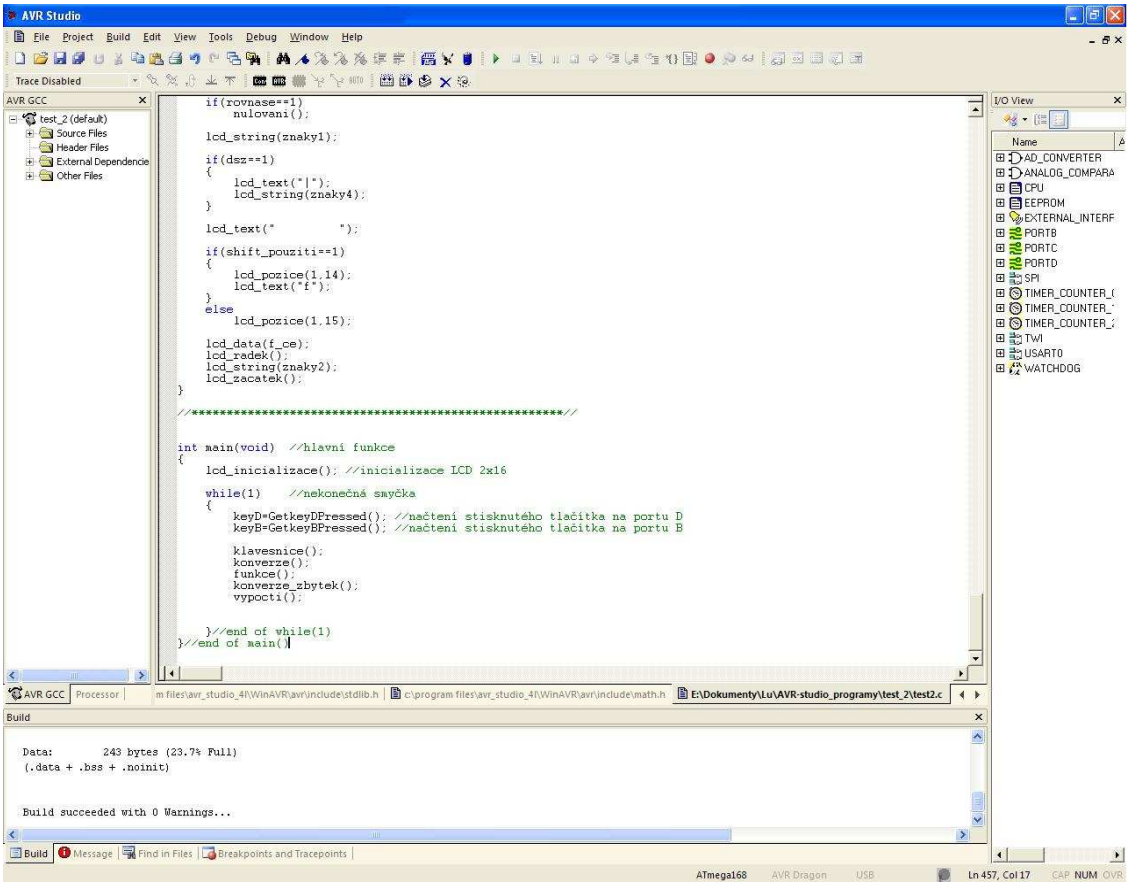
Pro zobrazování čísel zde slouží dvouřádkový displej z tekutých krystalů. Tyto displeje standardně obsahují řadiče a tak ke komunikaci stačí pouze 6 datových linek.

Napájení jsem zvolil kombinované, interní akumulátor, kvůli portabilitě, doplněnou o možnost připojení k externímu zdroji energie a tím šetření baterie (akumulátoru).

2. SOFTWARE

2.1. Vývojové prostředí

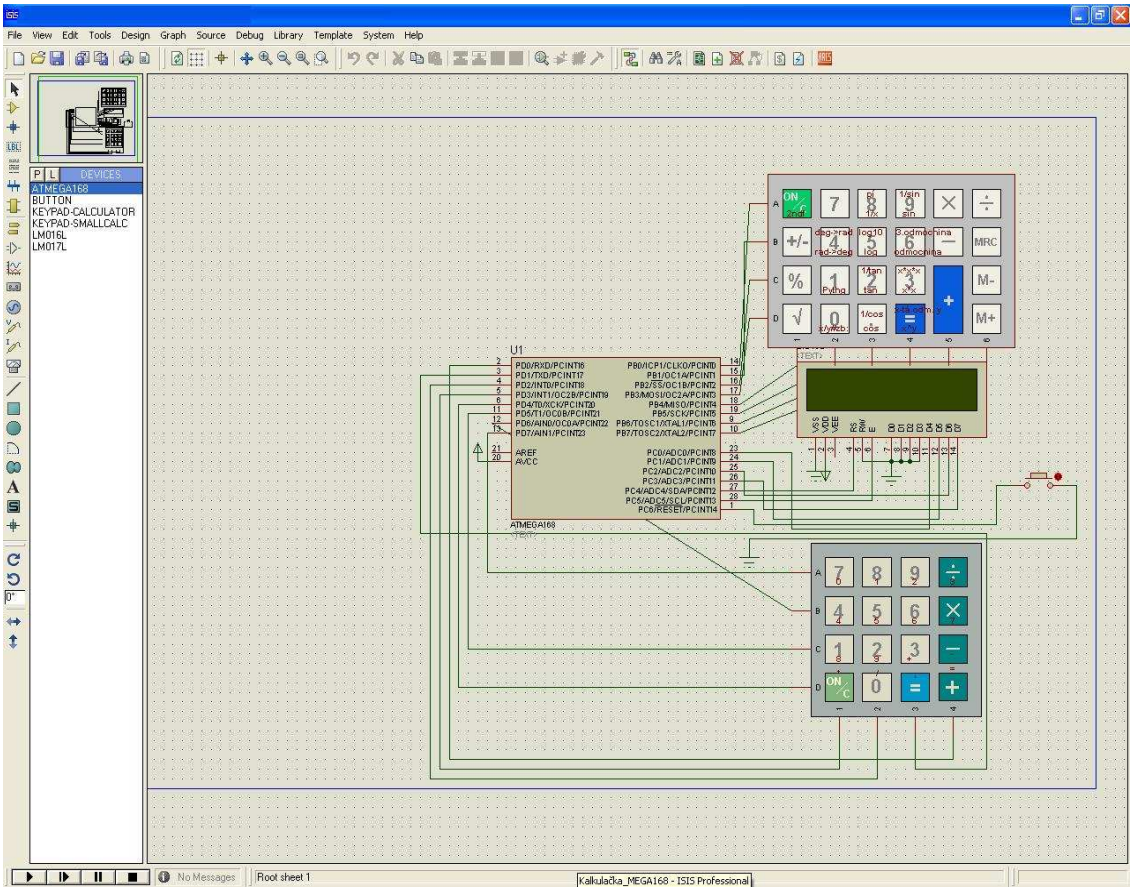
Pro vývoj programu jsem zvolil AVR Studio, které je zdarma k dispozici a vydává jej přímo firma Atmel. K samotnému textovému editoru lze doinstalovat i překládač, který pak poskytuje přímo HEX soubor. Ten je potřeba k naprogramování mikročipu.



Obrázek 1 - AVR Studio

2.2. Simulátor

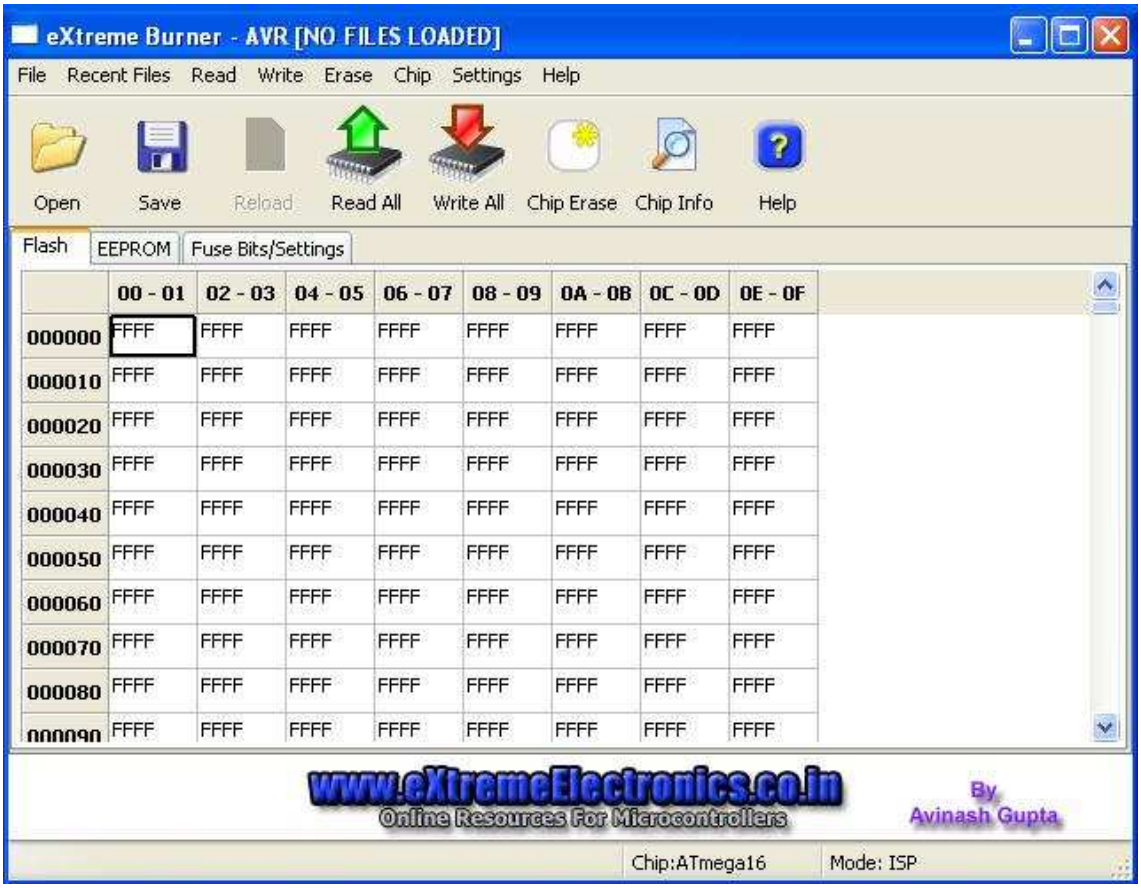
Pro ověření funkčnosti jsem před stavbou prototypu v nepájivém poli zvolil použití simulátoru ISIS. Ten je součástí balíku PROTEUS, který obsahuje kromě simulátoru také program ARES pro tvorbu DPS. Už základní verze simulátoru obsahuje velké množství součástek, nicméně je možné dostahovat další. Po sestavení obvodu a nahrání příslušného programu pro mikroprocesor pak stačí pouze zapnout simulaci a sledovat (ovládat) celé zapojení.



Obrázek 2 - Simulátor ISIS

2.3. Nahrání programu do mikroprocesoru

Pro „vypálení“ programu do mikročipu jsem použil program eXtreme Burner – AVR. Jeho ovládání je velice snadné. Možností tohoto programu je také programovat EEPROM nebo tzv. pojistky („fuses“). Program podporuje poměrně velké množství mikročipů. Jako výstup programu slouží USB port, na který je připojen USBasp programátor.

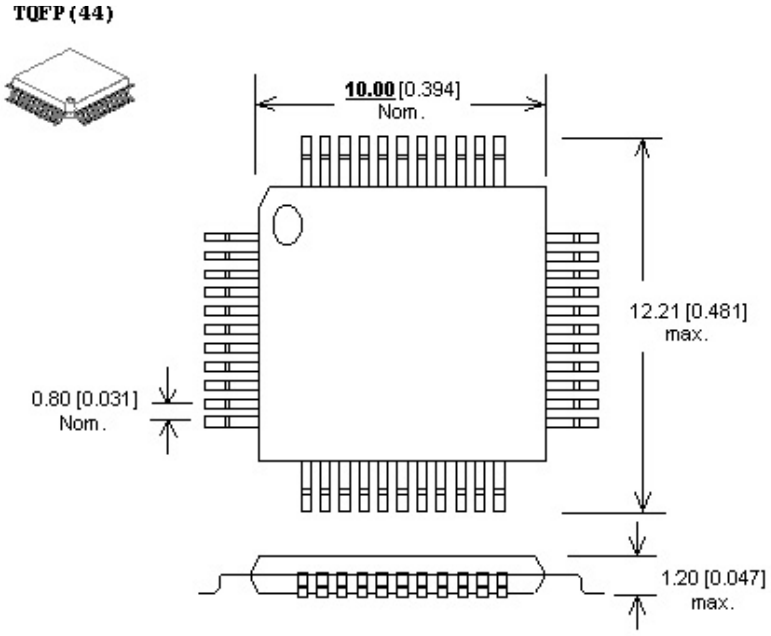


Obrázek 3 - eXtreme Burner - AVR

3. HARDWARE

3.1. Mikroprocesor

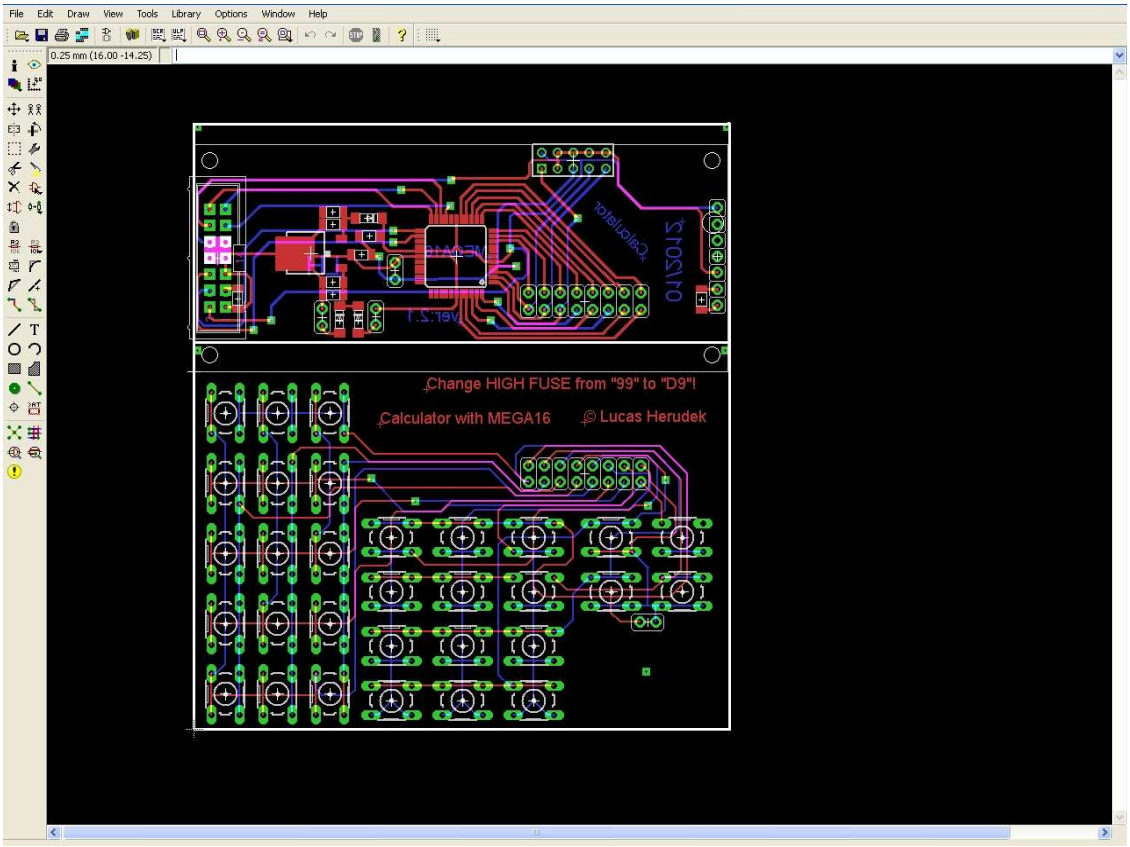
Základem tohoto zařízení je mikroprocesor, který řídí vše – od obsluhy klávesnice přes výpočet až po zobrazení údajů na LCD. Pro mé účely se nejlépe hodil ATMEGA168, ten jsem však nemohl sehnat v SMD provedení, dostupné bylo pouze pouzdro DIL28. To mi nevyhovovalo, protože je příliš velké a nevlezlo by se do krabičky. Proto jsem se rozhodl postavit celé zařízení na mikroprocesoru ATMEGA16A, který je snadněji dostupný v pouzdře TQFP44, které má rozměry pouze 12 x 12 x 1.2 mm, takže je velice malý. Má ovšem oproti ATMEGA168 o port více a tak tento port zůstal nevyužit.



Obrázek 4 - Pouzdro TQFP44

3.2. Deska plošných spojů

Desky jsem navrhoval v programu Eagle a vyrobil doma metodou tzv. „fotocesty“. Desky jsou oboustranné, protože použití propojek by bylo v tomto případě neúnosné. V tomto zařízení jsou celkem 2 DPS, jedna pro klávesnici a na druhé se nachází mikroprocesor. Tyto desky jsou propojeny klasickým plochým kabelem, známým třeba ze stolního počítače.



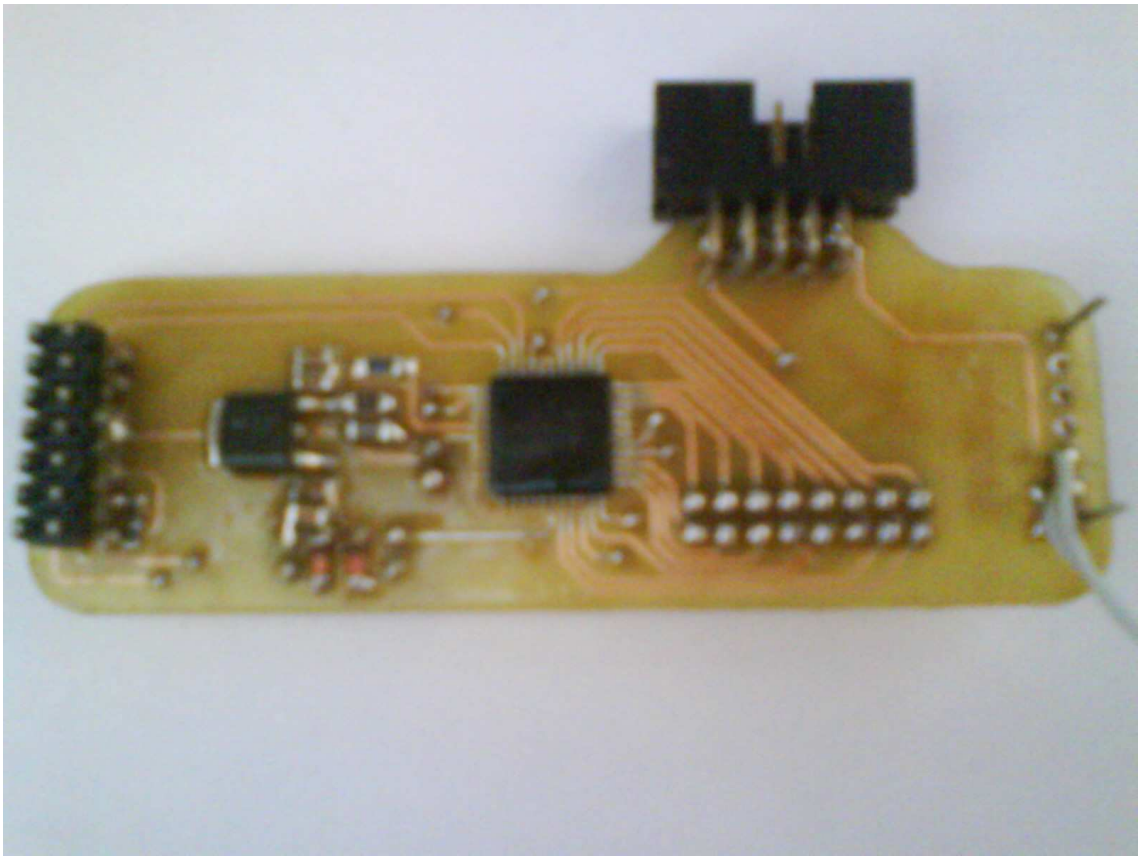
Obrázek 5 - Eagle-DPS

3.1.1. Klávesnice

Na desce klávesnice se nachází celkem 31 mikrospínačů a konektor. 32. tlačítko je vyvedeno kablíky a přes fastony připojeno k panelovému tlačítku, které je značně vetší než ostatní mikrospínače a slouží jako rovnítko (ENTER).

3.1.2. Hlavní DPS

Na této desce se nachází mozek kalkulátoru – mikroprocesor. Všechny součástky na této desce byly zvoleny pro povrchovou montáž (SMT). Tyto součástky ušetří spoustu místa a ušetří tak i peníze. Velikost je řady 1206, takže k zapájení není potřeba příliš-ných dovedností, stačí pevná ruka a správná pájka.

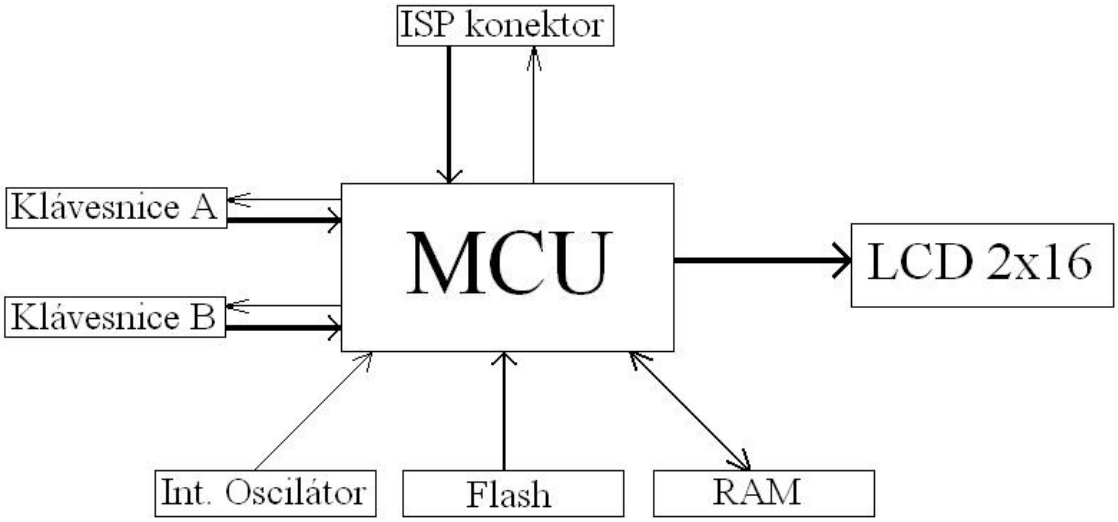


Obrázek 6 - Hlavní DPS

3.3. Blokové schéma

Toto schéma je vysvětlující, neslouží jako návod na stavbu, ale k pochopení principu funkce. Na tomto schématu lze jasně vidět, že mikroprocesor je hlavní součástka, která řídí vše.

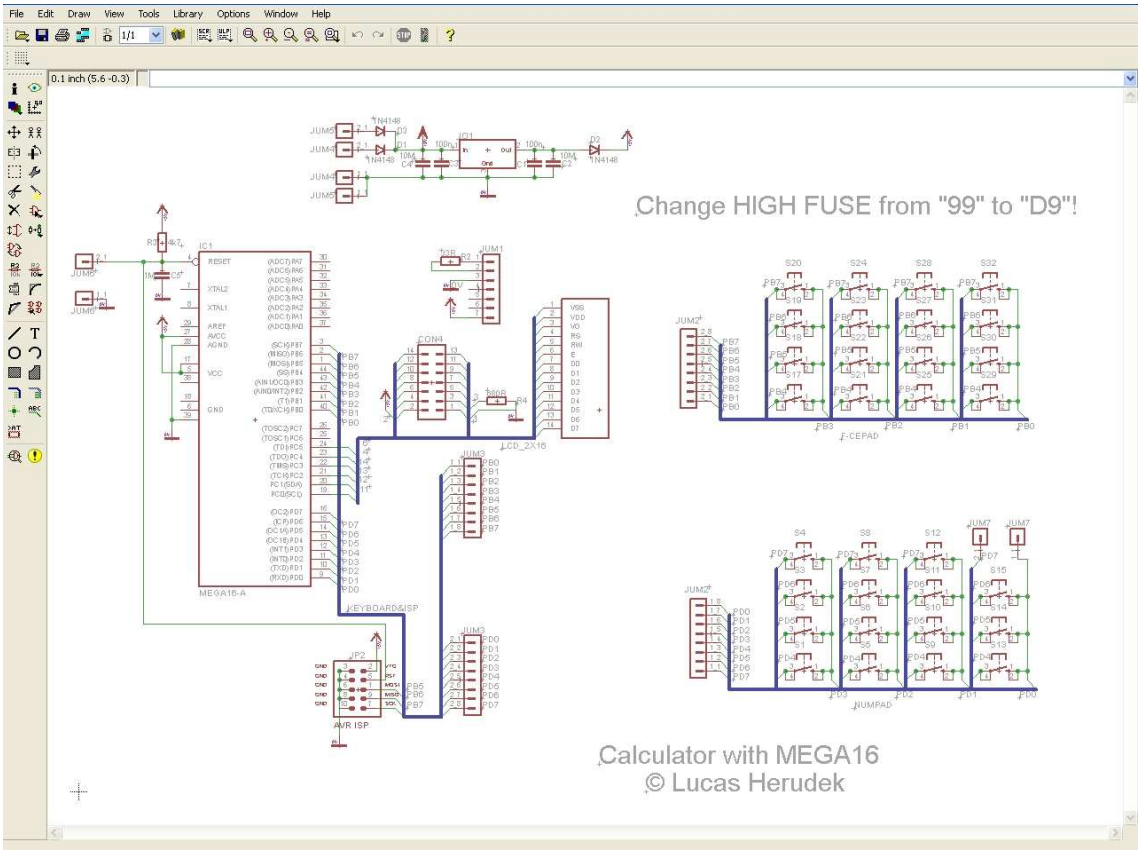
Vlevo lze vidět obě klávesnice, sloužící jako vstupy, nahoře je znázorněn ISP konektor sloužící k programování. Vpravo se nachází LCD. Všechny zmíněné části lze nalézt na desce plošných spojů. Oproti tomu si můžeme všimnout vnitřních součástí jako interního oscilátoru, flash paměti nebo RAM. Ty jsou zalisovány v pouzdře společně s jádrem.



Obrázek 7 - Blokové schéma

3.4. Schéma

Schéma celého zařízení bylo navrženo v programu Eagle. Největším prvkem je mikroprocesor. Dále je možno vidět kompletní stabilizační obvod nebo třeba zapojení klávesnice. Kvůli značnému počtu vodičů ve schématu jsem se rozhodl použít sběrnice a zpřehlednit tak schéma.



Obrázek 8 - Eagle-schéma

3.5. Krabička

Celé zařízení kalkulátoru je umístěno v černé plastové krabičce s rozměry: 25 x 94 x 128 mm, jejíž cena se pohybuje okolo 30 Kč. Na první pohled je nápadný LCD display na čelní stěně v horní části. Níže je umístěná klávesnice. Mezi nimi je tlačítko pro reset a celkový spínač/vypínač. Spínač pro zapínaní/vypínaní podsvícení displeje je umístěn na zadní stěně (krátké), kde se nachází jak programovací tak i napájecí konektor.

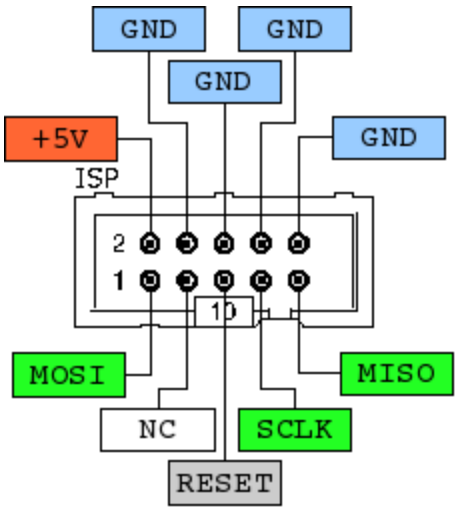
Obrázek 9 - Krabička

3.6. Programovací rozhraní

Mikroprocesor ATMEGA16 podporuje metodu tzv. ISP (In-System Programming), které má tu výhodu, že můžete programovat již postavené zařízení, bez potřeby

vyndávat mikročip a vkládat jej do speciálního programátoru. Toho jsem využil a osadil jsem ISP konektor přímo na základní desku. Tento konektor vyčnívá ven z krabičky, takže je velice snadné změnit softwarovou výbavu kalkulátoru.

Pro nahrání softwaru do mikroprocesoru je potřeba programátor podporující ISP. Já jsem použil USBasp. Tento programátor připojený k počítači převádí USB instrukce na ISP rozhraní, které je poté přímo připojeno na vývody mikročipu a rovnou tak programuje mikročip. K programování je zapotřebí celkem 6 linek: MOSI, MISO, SCK, RST, V_{cc} (+5 V nebo +3,3 V) a GND.



Obrázek 10 - ISP konektor

3.4.1. Pojistky („fuses“)

Mikroprocesor ATMEGA16 disponuje takzvanými pojistkami. Ty slouží ke změně základních funkcí a mnohdy právě špatné nastavení těchto pojistek zajistí nefunkčnost celého zařízení.

V tomto zařízení je nutné změnit horní pojistku z tovární hodnoty 0x99 na 0xD9. To zajistí vypnutí JTAG, které jinak neustále zajišťuje logickou 1 na některých vývodech portu 3 a způsobí nefunkčnost LCD.

4. PARAMETRY

4.1. Technické parametry

4.1.1. Rozměry (v x š x h)

Krabička	128 x 94 x 25	mm
Akumulátor	49 x 27 x 18	mm
LCD	85 x 30 x 15	mm
Aktivní část LCD	56 x 11	mm
DPS hlavní	39 x 85	mm
DPS klávesnice	56 x 85	mm
Kompletní zařízení	130 x 94 x 35	mm

4.2. Elektronické parametry

4.2.1. Napájení

Interní akumulátor	+8,4	V DC
Externí zdroj	+7-18	V DC
ISP konektor	+5	V DC
Výstup stabilizátoru (nevyvedeno)	+5	V DC
Napětí na mikroprocesoru a LCD	+5	V DC

4.2.2. Spotřeba

Celkově bez podsvícení	12,5 mA	±1 mA
Celkově s podsvícením	53,5 mA	±2 mA

4.2.3. Výdrž (interní akumulátor 200 mAh)

Bez podsvícení	až 16 hod
S podsvícením	až 3 hod 45 min

Vzhledem k tomu, že se jedná o akumulátor, výdrž je závislá na jeho nabití a stáří.

4.2.4. Softwarové parametry

Délka zadávaných čísel	až 15 cifer
Přesnost výpočtů	až 6 desetinných míst

5. POPIS PROGRAMU

5.1 Hlavní program (viz příloha)

Celý program jsem řádně okomentoval, aby byl lépe srozumitelný a přehledný pro případné úpravy.

5.1.1. Knihovny

Nejprve jsou definovány hlavičkové soubory (knihovny), obsahující ovladače portů, komunikaci s LCD a nebo třeba matematické funkce.

```
#include <stdio.h>              //vstup/výstup
#include <lcd.h>                //funkce pro LCD
#include <stdlib.h>             //práce se znaky a čísly (konverze...)
#include <math.h>               //matematické operace
#include <mega16.h>             //základní ovládání mikroprocesoru
```

Obrázek 11 - Knihovny

5.1.2. Definování proměnných

Posléze jsou definovány globální proměnné. Všechny proměnné v programu jsem definoval jako globální, více mi to vyhovovalo, jelikož používám jednu proměnnou ve více částech programu. A také kvůli přehlednosti, abych nemusel pořád hledat, kde a jak jsem proměnnou definoval, stačí se prostě podívat na začátek. V RAM paměti je zatím místa dost.

```
unsigned char rD,cD;            //znaky stisknutého řádku, sloupce  na portu D
unsigned char rB,cB;            //znaky stisknutého řádku, sloupce  na portu B

int tl_D;                       //pro rozpoznání stisknutého tlačítka na portu D
int tl_B;                       //pro rozpoznání stisknutého tlačítka na portu B

char f_ce;                      //znak početní funkce

int shift_pouziti;              //pro detekování použití druhé funkce (shift- tzv. 2ndf)

char znaky1[16];                //znaky pro 1. číslo
int index1=0;                   //pozice "kurzoru" ve znakovém poli pro 1. číslo

char znaky2[16];                //znaky pro 2. číslo
int index2=1;                   //pozice "kurzoru" ve znakovém poli pro 2. číslo

char vysledek_znaky[32];        //znaky pro výsledné číslo

double cislo1;                  //1.číslo
double cislo2;                  //2.číslo
double vysledek_cislo;          //vypočítané číslo

int pouziti_carky1=0;           //proměnná pro kontrolu použití desetinné čárky pro 1. číslo
int pouziti_carky2=0;           //proměnná pro kontrolu použití desetinné čárky pro 2. číslo

int rovnase=0;                  //pro tlačítko "rovná se"

int dsz=0;                      //pro dělení se zbytkem
long vysledek_celociselny;      //výsledek pro dělení se zbytkem
unsigned long zbytek;           //zbytek po dělení
char zbytek_znaky[16];          //znaky pro zbytek po dělení

int q;                          //slouží k nulování znaků
```

Obrázek 12 - Vložení knihoven

5.1.3. Klávesnice

Celá klávesnice se skládá ze dvou jedno-portových sub-klávesnic. Každá sub-klávesnice využívá celý port (8 pinů), a je zapojena do matice takže má každá 16 tlačítek (4*4). Obslužné programy jsou shodné pro oba porty (PORTB a PORTD), jen s malými rozdíly, proto zde popíšu princip funkce jen jednoho.

Nejdříve se nastaví 2. polovina portu na logickou 1 (řádky). Poté se nastaví tyto 4 piny jako vstup. V 1. polovině portu (sloupce) se piny nastaví jako výstupní. Tyto vstupy/výstupy však nejsou nikdy spuštěny najednou, ale postupně. Pozice vstupního a výstupního pinu hrají klíčovou roli při dekódování stisknuté hodnoty. Tyto pozice se přepnou celkem 3x (4 pozice) ne však společně, ale specificky. Nejdříve se 4krát přepne poloha výstupního pinu ve sloupci a až poté se vstupní pin v řádku přesune. To umožňuje rozeznat stisknuté tlačítko. Při stisku tlačítka pak dochází ke spojení dvou pinů a program si je automaticky rozliší a určí, které tlačítko bylo stisknuto a přiřadí mu příslušnou hodnotu (funkci).

```
unsigned char tl_D_hodnota()                //ovládání klávesice na portu D
{
    PORTD|= 0b00001111;                     //nastavení pull-upů

    for(cD=0;cD<4;cD++)                     //smyčka 3x přepni sloupce
    {
        DDRD&=~(0b01111111);                //1=výstup, 0=vstup (pozor, negace před závorkou!)
        DDRD|=(0b10000000>>cD);             //nastavení vstupů/výstupů pro dané sloupce a přesun pinu

        for(rD=0;rD<4;rD++)                 //smyčka 3x přepni řádky
        {
            if(!(PIND & (0b00001000>>rD)))  //kontrola a přesun vstupního pinu
            {
                switch(cD)                  //skládání hodnoty stisknutého tlačítka - sloupec
                {
                    case 0:tl_D=10;break;
                    case 1:tl_D=20;break;
                    case 2:tl_D=30;break;
                    case 3:tl_D=40;break;
                }

                switch(rD)                  //skládání hodnoty stisknutého tlačítka - řádek
                {
                    case 0:tl_D=tl_D+0;break;
                    case 1:tl_D=tl_D+1;break;
                    case 2:tl_D=tl_D+2;break;
                    case 3:tl_D=tl_D+3;break;
                }

                switch(tl_D)                //rozpoznávání hodnoty stisknutého tlačítka
                {
                    case 10:tl_D='+';break;
                    case 11:tl_D='*';break;
                    case 12:tl_D='/';break;
                    case 13:tl_D='=';break;
                    case 20:tl_D=',';break;
                    case 21:tl_D='3';break;
                    case 22:tl_D='6';break;
                    case 23:tl_D='9';break;
                    case 30:tl_D='0';break;
                    case 31:tl_D='2';break;
                    case 32:tl_D='5';break;
                    case 33:tl_D='8';break;
                    case 40:tl_D='#';break;
                    case 41:tl_D='1';break;
                    case 42:tl_D='4';break;
                    case 43:tl_D='7';break;
                    default:tl_D='X';break;
                }

                return tl_D;                //návrat hodnoty tl_D do proměnné tl_D_hodnota()
            }//konec if()
        }//konec for(rD)
    }//konec for(cD)

    return 0XFF;                            //Vrácení hodnoty 0XFF pokud nebylo stisknuto žádné tlačítko
}//konec unsigned char tl_D_hodnota()
```

Obrázek 13 - Ovládání klávesnice

5.1.4. Skládání čísel

Po stisku tlačítek a dekódování jejich hodnot je potřeba z nich složit číslo se kterým posléze můžeme počítat. Nejprve se hodnoty tlačítek vkládají do 16 místného znaku (15 cifer + zarážka). Jakmile je vyvolána konverze, tyto znaky složené dohromady se převedou číselnou proměnnou a je tak možné s ním klasicky počítat.

Na začátku programu se nejprve zjišťuje, zda je možno vložit znak do prvního čísla. 1. Číslo může mít maximálně 15 cifer. Pokud jich má méně, zjišťuje se stisknuté tlačítko a jeho hodnota se ukládá do 16 ciferného znaku a pozice cifry se posouvá. Pokud je delší, program čeká na zvolení funkce. Ty se pak dále rozpoznávají.

Pro možnost počítání desetinných čísel je zavedena funkce desetinné čárky. Ta může být v čísle přirozeně pouze jednou, je proto potřeba zabránit opakovanému vložení čárky do čísla.

Jako další jsou zde početní funkce. Ty zde přiřazují hodnotu tlačítka do proměnné, podle které je možno posléze provádět dané funkce. Zároveň ukončují zadávaní prvního čísla a přechází se k číslu druhému.

Stejně jako u prvního čísla, tak i u druhého se kontroluje délka. Pokud odpovídá požadavkům, stisknutá tlačítka se opět skládají do znaků a posléze konvertují. Toto pokračuje, dokud není stisknut ENTER nebo dokud se čísla vlezou.

```
void klavesnice() //obsluha klávesnice
{
/*  SKLÁDÁNÍ 1.-ho ČÍSLA    */
    if(index1<=15)                                          //pokud je cifer 1.-ho čísla méně než 16,
    {                                                       //pokračuj v zadávání 1.-ho čísla
        switch(tl_D)                                        //rozpoznej stisknuté tlačítko
        {
            case '.':   if(index1<=13 && pouziti_carky1==0) //kontrola možnosti vložit des. čárku
                        {
                            znaky1[index1]='.';             //vložení desetinné čárky pro první číslo
                            index1++;
                            pouziti_carky1=1;               //zajištění nevložení 2 des. čárek do 1.-ho čísla
                        }break;//konec if(index1<=13 && pouziti_carky1==0)
            case '+': f_ce='+'; index1=16;break;            //stisknuto +, sečti, konec zadávání 1.-ho čísla
            case '#': index1--;znaky1[index1]=' ';break;    //stisk # slouží jako backspace
            case '*': f_ce='*'; index1=16;break;            //stisknuto *, vynásob, konec zadávání 1.-ho čísla
            case '/': f_ce='/'; index1=16;break;            //stisknuto /, poděl, konec zadávání 1.-ho čísla
            default : f_ce='_';break;                       //nic nestiskuto
        }//konec switch(tl_D)

        if(index1<15)                                       //pokud je cifer 1.-ho čísla méně než 15,
        {                                                   //pokračuj v zadávání 1.-ho čísla
            switch(tl_D)                                    //skládání 1.-ho čísla
            {
                case '0': znaky1[index1]='0';index1++;break;//vlož do řetězce znaků a posuň pozici
                case '1': znaky1[index1]='1';index1++;break;
                case '2': znaky1[index1]='2';index1++;break;
                case '3': znaky1[index1]='3';index1++;break;
                case '4': znaky1[index1]='4';index1++;break;
                case '5': znaky1[index1]='5';index1++;break;
                case '6': znaky1[index1]='6';index1++;break;
                case '7': znaky1[index1]='7';index1++;break;
                case '8': znaky1[index1]='8';index1++;break;
                case '9': znaky1[index1]='9';index1++;break;
            }//konec switch(tl_D)
        }//konec if(index1<15)
    }//konec if(index1<=15)
```

Obrázek 14 - Skládání prvního čísla

5.1.5. Funkce

Tato část programu obsahuje to hlavní – funkce kalkulátoru. Tento úsek jsem napsal co možná nejpřehledněji, aby se v něm vyznal i méně znalý uživatel a mohl si jej upravit pro vlastní potřeby přidáváním libovolných funkcí.

V programu je zavedena takzvaná „druhá funkce“, je tak možné získat mnohem více funkcí. Teoreticky je možné zavést „třetí funkci“ jako druhou funkci druhé funkce. Toho jsem zatím nevyužil, ale nevylučuji to v budoucnu.

Na začátku této části programu se kontroluje, zda byla zvolena druhá funkce. Poté se přechází přímo k výběru funkcí dle stisknutého tlačítka.

```
void funkce()
{
    if(shift_pouziti==0)                                                    //kontrola nepoužití 2ndf
    {
        switch(f_ce)                                                        //rozpoznání funkcí
        {
            case '+': vysledek_cislo = cislo1 + cislo2;break;               //sčítání
            case '-': vysledek_cislo = cislo1 - cislo2;break;               //odčítaní
            case '*': vysledek_cislo = cislo1 * cislo2;break;               //násobení
            case '/': vysledek_cislo = cislo1 / cislo2;break;               //dělení
            case 'E': vysledek_cislo = pow(cislo1,cislo2);break;            //x na y
            case 'M': vysledek_cislo = pow(cislo1,2);rovnase=1;break;       //x na druhou
            case 'I': vysledek_cislo = sqrt(cislo1);rovnase=1;break;        //2. odmocnina
            case 'P': vysledek_cislo = sin(cislo1*(M_PI/180));rovnase=1;break;  //sinus
            case 'L': vysledek_cislo = cos(cislo1*(M_PI/180));rovnase=1;break;  //cosinus
            case 'H': vysledek_cislo = tan(cislo1*(M_PI/180));rovnase=1;break;  //tangens
            case 'J': vysledek_cislo = log10(cislo1);rovnase=1;break;       //logaritmus se základem 10
            case 'F': vysledek_celociselny = (long)cislo1/(long)cislo2;
                      zbytek = (unsigned long)cislo1%(unsigned long)cislo2;
                      dsz=1;break;                                          //dělení se zbytkem
            case 'C': vysledek_cislo = hypot(cislo1, cislo2);;break;        //Pyth.věta-výpočet přepony
            case 'N': vysledek_cislo = cislo1*(180/M_PI);rovnase=1;break;   //převod radiánů na stupně
            case 'B': if(cislo2==0) {vysledek_cislo=-cislo1;rovnase=1;}
                      else {znaky2[0]='-';f_ce='_';}break;                  //změna znaménka
            case 'O': vysledek_cislo = sinh(cislo1);rovnase=1;break;        //hyperbolický sin
            case 'K': vysledek_cislo = cosh(cislo1);rovnase=1;break;        //hyperbolický cos
            case 'G': vysledek_cislo = tanh(cislo1);rovnase=1;break;        //hyperbolický tan
            case 'A': f_ce='_';shift_pouziti=1;break;                       //2ndf
        }//konec switch(f_ce)
    }//konec if(shift_pouziti==0)

    else if(shift_pouziti==1)                                               //2nd funkce
    {
        switch(f_ce)
        {
            case 'E': vysledek_cislo = pow(cislo1, 1/cislo2);break;         //x-tá odmocnina y
            case 'M': vysledek_cislo = cislo1*cislo1*cislo1;rovnase=1;break;    //x na třetí
            case 'I': vysledek_cislo = pow(cislo1,0.3333333333);rovnase=1;break;//3.odmocnina
            case 'P': vysledek_cislo = asin(cislo1)*(180/M_PI);rovnase=1;break; //sin na -1
            case 'L': vysledek_cislo = acos(cislo1)*(180/M_PI);rovnase=1;break; //cos na -1
            case 'H': vysledek_cislo = atan(cislo1)*(180/M_PI);rovnase=1;break; //tan na -1
            case 'J': vysledek_cislo = log(cislo1);rovnase=1;break;         //přirozený logaritmus
            case 'K': vysledek_cislo = 3.1415926535;rovnase=1;break;        //pi
            case 'B': vysledek_cislo = cislo1*pow(10,cislo2);;break;        //exp: číslo *10^x
            case 'F': vysledek_cislo = 1/cislo1;rovnase=1;break;            //převrácená hodnota
            case 'N': vysledek_cislo = cislo1*(M_PI/180);rovnase=1;break;   //převod stupňů na radiány
            case 'O': vysledek_cislo = exp(cislo1);rovnase=1;break;         //exponencionální funkce
            case 'C': cislo2=-cislo2*cislo2;cislo1=pow(cislo1,2);
                      vysledek_cislo = sqrt(cislo1+cislo2);break;           //Pyth.věta-výpočet ramene
            case 'G': vysledek_cislo = atan2(cislo1, cislo2);break;         //tan(x/y) na -1
            case 'A': srandom(cislo1);vysledek_cislo = random();rovnase=1;break;//pseudonáhodné číslo
            default:  vysledek_cislo = 0;break;
        }//konec switch(f_ce)
    }// konec else if(shift_pouziti==1)
}//konec void funkce()
```

Obrázek 15 - Funkce

5.1.6. Konverze

Jak již bylo řečeno, pro počítání potřebujeme konvertovat znaky složené do celků na číselné proměnné. Toto se děje v této části programu. Pro tuto konverzi existují přímo funkce, které toto provedou automaticky.

```
void konverze()
{
    cislo1=strtod(znaky1, &endptr);      //konverze znaků1 na 1. číslo
    cislo2=strtod(znaky2, &endptr);      //konverze znaků2 na 2. číslo
    /*  *endptr se nachází v knihovně stdlib.h  */
}//konec void konverze()
```

Obrázek 16 - Konverze

5.1.7. Nulování

Důležitou součástí hlavního programu je nulování. To je potřebné k opakovanému počítání, jinak řečeno, aby bylo možné dále počítat s právě vypočteným výsledkem.

Nejprve se vloží takzvaná „zarážka“ na konec znaků pro druhé číslo. Dále se kontroluje hodnota proměnné pro dělení se zbytkem (rozhodovací proměnná) a provedou se příslušné příkazy. Pokud je hodnota rovna dvěma, vynuluje se. Pokud je rovna jedné, znamená to, že bylo použito dělení se zbytkem a proto je potřeba tento výsledek nahrát do proměnné s výsledkem a konvertovat ji na znaky, ovšem bez desetinných míst. Poté se hodnota rozhodovací proměnné zvýší na 2. A pokud je obsah rozhodovací proměnné jiný než 1 nebo 2 (tzn., že není použito děleno se zbytkem), provede se standardní konverze čísla na znaky se šesti desetinnými místy. Poté se všechny důležité proměnné nulují. Zajímavé je možná nulování znaků pro druhé číslo, které se provádí přepisem původních čísel mezerami. Toto se provádí 16x. Následuje už jen vymazání displeje a příkaz pro počkání 50 milisekund.

```
void nulovani()                     //nulování proměnných pro opakované počítání
{
    znaky2[index2]='\0';            //vlož zarážku

    if(dsz==2)                      //pokud je proměnná rovna dvěma
        dsz=0;                      //vynuluj ji

    if(dsz==1)                      //pokud je použito dělení se zbytkem
    {
        cislo1=vysledek_celociselny;//ulož celočíselný výsledek do prvního čísla
        dtostrf(cislo1,0,0,znaky1); //překonvertuj první číslo na znaky bez desetinných míst
        dsz=2;
    }
    else                            //pokud není použito dělení se zbytkem
    {
        cislo1=vysledek_cislo;      //ulož reálný výsledek do prvního čísla
        dtostrf(cislo1,0,6,znaky1); //překonvertuj první číslo na znaky se 6 desetinnými místy
    }

    vysledek_cislo=0;               //vynuluj reálný výsledek
    vysledek_celociselny=0;         //vynuluj celočíselný výsledek
    zbytek=0;                       //vynuluj zbytek
    pouziti_carky2=0;               //vynuluj kontrolu použití des. čárky v druhém čísle
    shift_pouziti=0;                //vynuluj použití 2ndf
    rovnase=0;                      //vynuluj stisknutí tlačítka rovná se
    f_ce='_';                       //vynuluj funkci
    cislo2=0;                       //vynuluj druhé číslo
    dtostrf(cislo2,0,0,znaky2);     //vynuluj druhé číslo překonvertuj druhé číslo na znaky
    index2=1;                       //znaky se budou zapisovat až od druhé pozice (na první se vkládá mínus)

    for(q=-1;q<=15;q++)
        znaky2[q]=' ';              //16x vlož mezeru do znaků pro druhé číslo

    znaky2[0]=' ';                  //přepiš první pozici ve druhých znacích na mezeru

    lcd_vymaz();                    //vymaž LCD
    _delay_ms(50);                  //počkej 50 milisekund
}//konec void nulovani()
```

Obrázek 17 - Nulování

5.1.8. Výpis na LCD

Tato část zajišťuje výpis na displej. K tomu je zapotřebí znakové proměnné, která se pomocí příkazu zobrazí na displeji. Proto musí opět proběhnout konverze, tentokrát však z čísla na znaky. Při konverzi lze zvolit i počet desetinných míst (přesnost).

Hned na úvod je vypsáno první číslo, které při opakovaných výpočtech slouží jako výsledek. Následuje jej kontrola hodnoty proměnné indikující dělení se zbytkem. Pokud je tato hodnota rovna jedné, celočíselný výsledek a zbytek po dělení se konvertují na znaky. Tento zbytek po dělení je následně vypsán za svislou čáru. Poté se pozice kurzoru na LCD přesune na patnáctou pozici v prvním řádku a kontroluje se, zda byla zvolena druhá funkce. Pokud ano, vypíše se zde malé f. Pokud ne, vloží se zde mezera. Následně se kurzor přesouvá na šestnáctou pozici, kde se vypíše zvolená funkce jako velké písmeno. Následně se kurzor přesouvá na začátek druhého řádku, kde se vypíše druhé číslo. Nakonec se kurzor přesouvá na úplný začátek LCD.

```
void vypis()                    //zajišťuje výpis proměnných
{
    lcd_string(znaky1);         //vypiš 1. číslo

    if(dsz>=1)                  //poukud se dělí se zbytkem, vypiš zbytek za svislou čáru
    {
        if(dsz==1)                                          //pokud je použito dělení se zbytkem
        {
            dtostrf(vysledek_celociselny,0,0, vysledek_znaky);  //kunvertuj celočíselný výsledek na znaky
            dtostrf(zbytek,0,0,zbytek_znaky);               //konvertuj zbytek po dělení na znaky
        }//konec if(dsz==1)
        lcd_text("|");          //zobrazení svislé čáry
        lcd_string(zbytek_znaky);//výpis zbytku
        lcd_text("          "); //vypiš několik mezer, pro přepsání předchozího čísla
    }

    lcd_pozice(1,14);           //nastav kurzor na 14. pozici v prvním řádku

    if(shift_pouziti==1)        //pokud je použita 2ndf, vypiš "f"
        lcd_text("f");          //vypiš "f"
    else                        //jestli není použito 2ndf
        lcd_text(" ");
    lcd_pozice(1,15);           //nastav kurzor na 15. pozici v prvním řádku
    lcd_data(f_ce);             //vypiš používanou funkci (+;-,*,...)
    lcd_radek();                //přesuň kurzor na druhý řádek
    lcd_string(znaky2);         //vypiš 2. číslo
    lcd_zacatek();              //přesuň kurzor na začátek (1. řádek, 1. pozice)
}
```

Obrázek 18 - Výpis na LCD

5.1.9. Hlavní část programu (main)

Hlavní část programu se skládá ze dvou částí, jedenkrát provedených příkazů a nekonečné smyčky.

V první části se příkazy provedou pouze jednou a to bezprostředně po spuštění kalkulátoru.

Nejprve se vloží mezera na první pozici druhých znaků. Následně se takzvaně inicializuje LCD. Toto je zapotřebí kvůli řadiči ovládajícím jednotlivé pixely displeje. Tento příkaz obsahuje například způsob komunikace (4 nebo 8 bitové). Toto celé se provádí pouze jednou.

Druhá část programu je uzavřena do takzvané „nekonečné smyčky“. Provedení příkazů uvnitř se neustále opakuje od spuštění kalkulátoru až do jeho vypnutí. Tuto smyčku nelze softwarově přerušit.

V této části nejprve probíhá načtení hodnot stisknutých tlačítek. Poté se čeká 20 milisekund. To zabraňuje vložení více znaků v krátké chvíli. Následně se již pouze volají různé části programu. Snad netřeba zdůrazňovat, že jejich pořadí je klíčové pro správnou funkci kalkulátoru. Jako první se volá podprogram obsluhující klávesnici. Následně se tyto hodnoty zkonvertují. Poté se rozliší použitá početní funkce. Následuje podmínka, pokud je stisknuto rovná se a je zvolena funkce, program přejde na nulování. Nakonec se vše zobrazí na displeji a celá smyčka se opakuje.

```
int main(void)                          //hlavní funkce
{
    znaky2[0]=' ';                      //vlož mezeru na 0. pozici ve druhém čísle

    lcd_inicializace();                 //inicializace LCD 2x16

    while(1)                            //nekonečná smyčka
    {
        tl_D=tl_D_hodnota();            //načti stisknuté tlačítka na portu D
        tl_B=tl_B_hodnota();            //načti stisknuté tlačítka na portu B
        _delay_ms(20);                  //počkej 20 milisekund

        klavesnice();                   //slož jednotlivé znaky do celku
        konverze();                     //překonvertuj celek znaků na číslo
        funkce();                       //vypočítej

        if(rovnase==1 && f_ce!='_') //pokud je stisknuto rovná se a je zvolena funkce, přejdi na nulování
            nulovani();

        vypis();                        //výsledek vypiš na LCD
    }//konec while(1)
}//konec main()
```

Obrázek 19 - Hlavní část programu

5.2 Matematické funkce (math.h)

Tato knihovna obsahuje kromě různých konstant hlavně matematické funkce. Příkazy jsou pojmenovány dle názvu prováděné operace (například sin je funkce sinus), lze proto snadno odhadnout, k čemu příkaz slouží. Knihovna je volně dostupná na internetu a lze ji také samozřejmě rozšířit o vlastní funkce.

```
extern double exp(double __x) __ATTR_CONST__;
#define expf    exp     /**< The alias for exp(). */

/**
    The cosh() function returns the hyperbolic cosine of \a __x.
 */
extern double cosh(double __x) __ATTR_CONST__;
#define coshf   cosh        /**< The alias for cosh(). */

/**
    The sinh() function returns the hyperbolic sine of \a __x.
 */
extern double sinh(double __x) __ATTR_CONST__;
#define sinhf   sinh        /**< The alias for sinh(). */

/**
    The tanh() function returns the hyperbolic tangent of \a __x.
 */
extern double tanh(double __x) __ATTR_CONST__;
#define tanhf   tanh        /**< The alias for tanh(). */

/**
    The acos() function computes the principal value of the arc cosine of
    \a __x.  The returned value is in the range [0, pi] radians. A domain
    error occurs for arguments not in the range [-1, +1].
 */
extern double acos(double __x) __ATTR_CONST__;
#define acosf   acos        /**< The alias for acos(). */

/**
    The asin() function computes the principal value of the arc sine of
    \a __x.  The returned value is in the range [-pi/2, pi/2] radians. A
    domain error occurs for arguments not in the range [-1, +1].
 */
extern double asin(double __x) __ATTR_CONST__;
#define asinf   asin        /**< The alias for asin(). */

/**
    The atan() function computes the principal value of the arc tangent
    of \a __x.  The returned value is in the range [-pi/2, pi/2] radians.
 */
extern double atan(double __x) __ATTR_CONST__;
#define atanf   atan        /**< The alias for atan(). */

/**
    The atan2() function computes the principal value of the arc tangent
    of <em>__y / __x</em>, using the signs of both arguments to determine
    the quadrant of the return value.  The returned value is in the range
    [-pi, +pi] radians.
 */
extern double atan2(double __y, double __x) __ATTR_CONST__;
#define atan2f  atan2       /**< The alias for atan2(). */

/**
    The log() function returns the natural logarithm of argument \a __x.
 */
extern double log(double __x) __ATTR_CONST__;
#define logf    log     /**< The alias for log(). */
```

Obrázek 20 – Ukázka knihovny math.h

5.3 Ovládání LCD (lcd.h)

Tuto knihovnu jsem stáhnul z internetu a upravil pro mé použití. Obsahem jsou definice funkcí a příkazů pro ovládání LCD včetně příkazu pro inicializaci LCD, stačí tak pouze jeden příkaz a celá inicializace sestávající z více příkazu se provede. Programátorovi to tak značně usnadňuje práci.

```
void lcd_vymaz( void )
{
    lcd_prikaz( 0x01 ); // volání funkce pro odesílání instrukcí
    _delay_ms( 2 );          // čas pro vykonání instrukce
    lcd_zacatek();
}

void lcd_inicializace( void )
{
    // nastavení pinů procesoru, které jsou připojeny k LCD displeji jako výstupní porty
    uint8_t pins = (0x0F << PD0) |  // datové vodice
                   (1<<RS) |        // port RS
                   (1<<E);      // port ENABLE
    DDRC |= pins;

    // počáteční nastavení všech výstupů na nulu
    PORTC &= ~pins;

    // čekání na komunikaci s LCD po připojení napájení
    _delay_ms(15);

    // "reset" displeje a následné trojnásobné potvrzení ENABLE pulsem
    lcd_out( 0x30 );
    _delay_ms( 1 );

    lcd_enable();
    _delay_ms( 1 );

    lcd_enable();
    _delay_ms( 1 );

    // inicializace 4-bitového režimu komunikace
    lcd_out( 0x20 );
    _delay_ms( 5 );

    // potvrzení 4-bitového módu, nastavení dvouřádkového displeje a fontu s rozměrem 5x8 bitů
    lcd_prikaz( 0x28 ); // (0b001010xx)
    _delay_us(40);

    // zapnutí displeje, vypnutí kurzoru a blikání
    lcd_prikaz( 0x0C ); // 0x0F pro blikající kurzor

    // nastavení posuvu kurzoru vpravo a vypnutí posouvání textu
    lcd_prikaz( 0x06 );

    // volání funkce pro vymazání displeje
    lcd_vymaz();

    //vložení vlastních znaků
    vlastni_znaky();
}

void lcd_zacatek( void )
{
    lcd_prikaz( 0x02 );
    _delay_ms( 2 );
}
```

Obrázek 21 – Ukázka knihovny lcd.h

6. ZÁVĚR

Podařilo se mi vytvořit plně funkční kalkulátor, který je v praxi využitelný zejména ve školách a podobných institucích. Hlavní předností tohoto zařízení je možnost vytvoření vlastních vzorců a tím také značnému usnadnění práce. To jsem využil a vytvořil funkci pro výpočet odvěsny v pravoúhlém trojúhelníku s pomocí Pythagorovy věty.

Vývoj softwaru trval čtyři měsíce. Problémy se objevily hned ze začátku, a sice u výpisu znaků na LCD. Standardní a velice známá funkce printf nefungovala. Jelikož bylo potřeba tento problém rychle vyřešit, použil jsem funkci z knihovny lcd.h, která neposílá text jako celek, ale jednotlivá písmena a znaky samostatně. Nakonec se toto řešení ukázalo jako uspokojivé, bylo avšak zapotřebí dalších konverzí.

Stavba a zprovoznění kalkulátoru mi zabralo asi jeden a půl měsíce. Nejobtížnější bylo správné a přesné vyvrtání otvorů do krabičky a následné přidání popisků.

Cena materiálu se pohybuje okolo 500 Kč. Veškeré komponenty jsou snadno dostupné a dají se koupit v několika různých obchodech.

Jako další vylepšení je možno zavést ukládání čísel nebo třeba vkládání textu pomocí klávesnice a jeho následné uložení do paměti EEPROM.

7. SEZNAM POUŽITÝCH ZDROJŮ

7.1. Obrázky

Aquaticus. AVR ISP konektor.

http://aquaticus.info/avr

Tipa. Krabička Z19-KP5.

http://www.tipa.eu/cz/krabicka-z-19-kp5/d-85481/

Weilei. Pouzdro TQFP44.

http://www.weilei.cz/tqfp44.php

7.2. Programové části

The Open Group. Funkce pseudonáhodného čísla (random).

http://pubs.opengroup.org/onlinepubs/009695399/functions/rand.html

The Open Group. Funkce konverze znaků na číslo.

http://pubs.opengroup.org/onlinepubs/007904975/functions/strtod.html

The Open Group. Knihovna pro práci se znaky a čísly (stdlib.h).

http://pubs.opengroup.org/onlinepubs/007904975/basedefs/stdlib.h.html

Engbedded Atmel AVR® Fuse Calculator. Výpočet pojistek („fuses“).

http://www.engbedded.com/fusecalc/

Wikipedie. Obsah knihovny pro matematické funkce (math.h).

http://cs.wikipedia.org/wiki/Math.h

Ing. Vladimír Dumek, Ph.D. Programování v jazyku C.

http://drogo.fme.vutbr.cz/~jroupec/ccpp/pdf/prog_v_jazyku_C.pdf

EXTREME ELECTRONICS. Teorie a ukázka maticových klávesnic.

http://extremeelectronics.co.in/avr-tutorials/4x3-matrix-keypad-interface-avr-tutorial/

DharmaniTech. 4x4 Matrix Key-board Interfacing with ATmega32.

http://www.dharmanitech.com/2008/11/4x4-matrix-key-board-interfacing-with.html

Schmija2 – MAM wiki. Obsah a popis knihovny ovládající LCD (lcd.h).

http://noel.feld.cvut.cz/vyu/a2m99mam/index.php/U%C5%BEivatel:Schmija2

7.3. Datasheety

APEX SCIENCE & ENGINEERING CORP. Datasheet od LCD.

http://www.volny.cz/hezky.den/datasheet/RC162021YFHLYB.pdf

ATMEL. Datasheet ATMEGA16A.

http://doc.gmecdn.cz/958/958-176/dsh.958-176.1.pdf

ATMEL. Datasheet ATMEGA168.

http://doc.gmecdn.cz/432/432-192/dsh.432-192.1.pdf

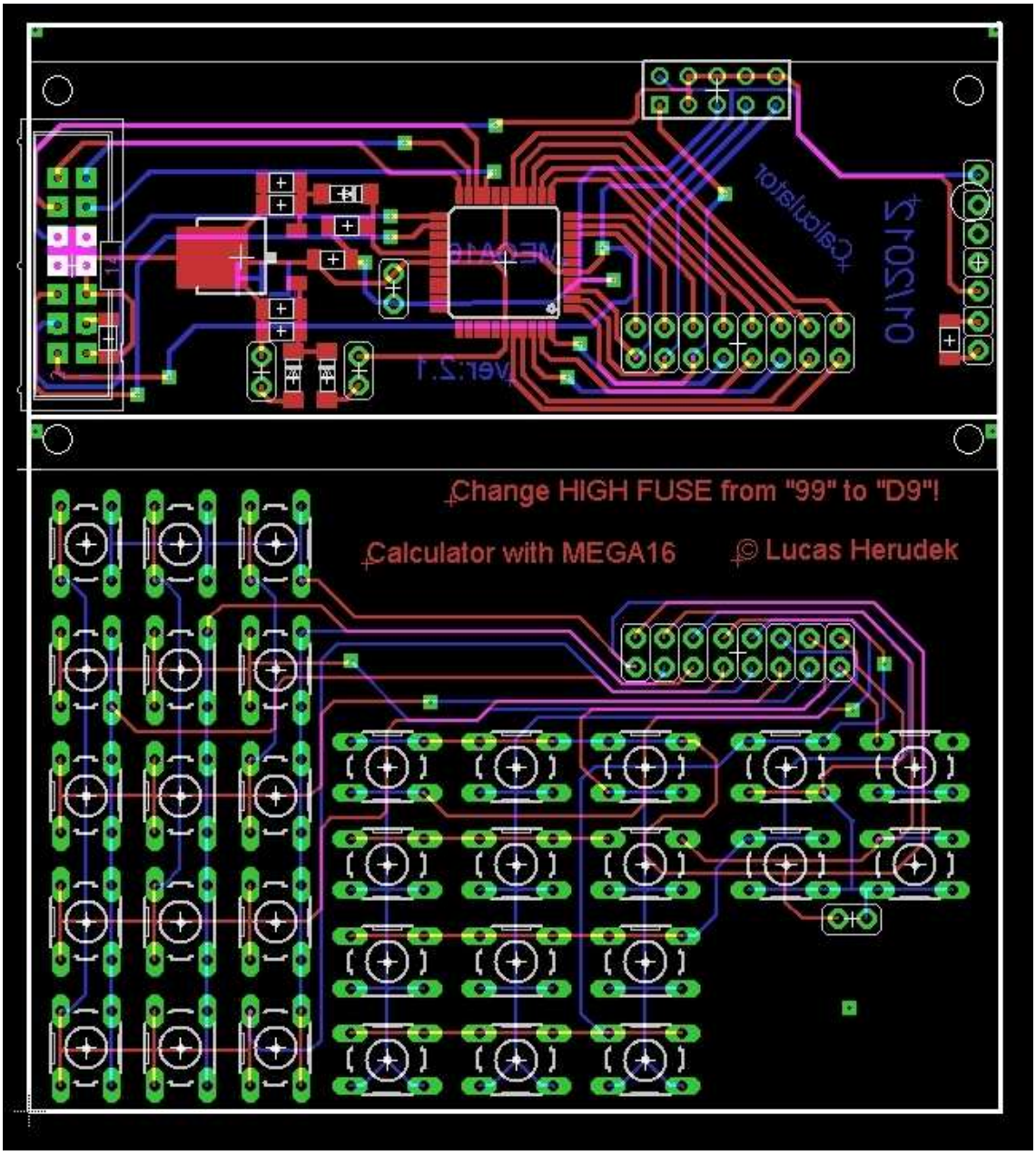
8. PŘÍLOHY

8.1. Fotodokumentace

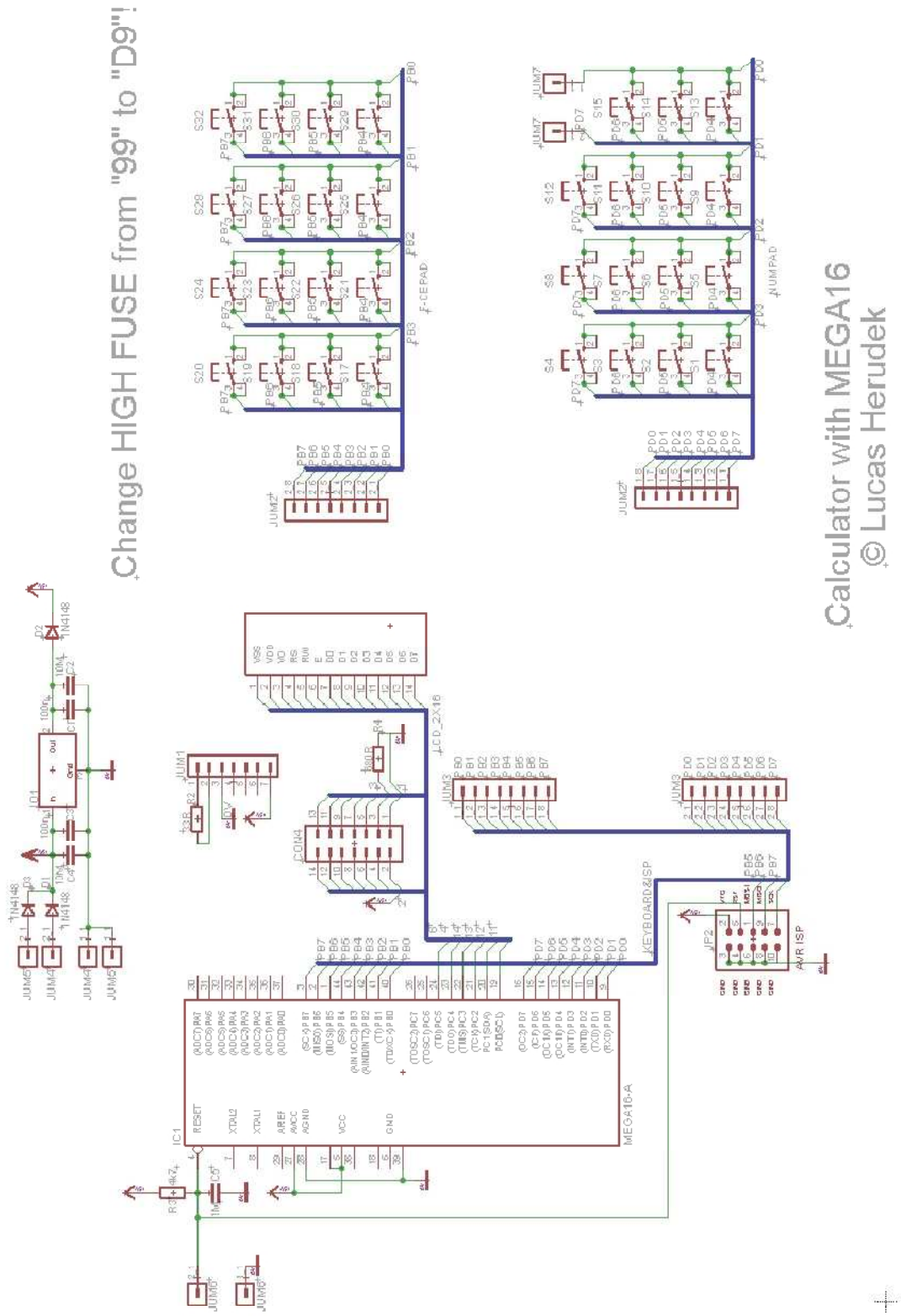
Obrázek 22 - Pohled zepředu

Obrázek 23 - Pohled zleva

Obrázek 24 - Pohled shora



Obrázek 25 – DPS



Obrázek 26 – Schéma

8.2. Hlavní program
- viz příloha 1 (Maticová klávesnice)

8.3. Knihovna pro ovládání LCD (lcd.h)
– viz příloha 2

8.4. Knihovna matematických funkcí (math.h)
– viz příloha 3