Programování

1	**SPŠ Chomutov**	**Kapusňák Jiří**
4.1.2022	**Světelný řádek**	**V4**

Zadání

Vytvořte pomocí světelného řádku a mikrokontrolér ATMega128 program pro funkční hodiny.

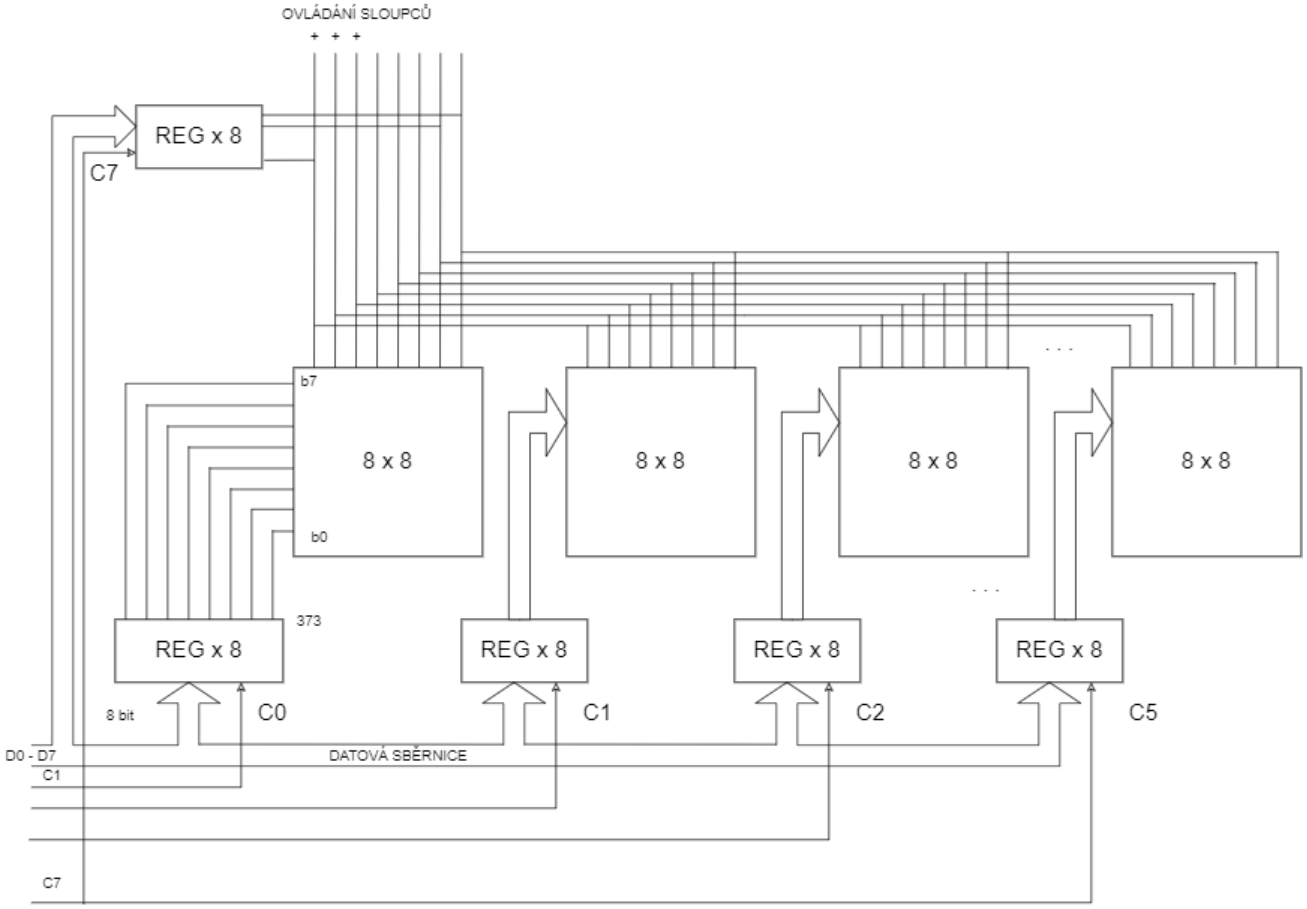
Teorie

Mikrokontrolér propojíme se světelným řádkem, kde PORTB propojíme se 7 piny registrů a PORTD s 8 datovými piny, počítač propojíme s mikrokontrolérem pomocí programátoru JTAG. Dále budeme potřebovat software, který nám překompiluje náš kód do kódu zdrojového, v našem případě Programmers Notepad.

K našemu projektu musíme vytvořit MakeFile, kde uchováváme veškeré nastavení našeho mikrokontroléru, ten vytvoříme pomocí programu mfile. Do MakeFilu musíme nastavit náš správny procesor (atmega128), námi zvolený programátor (jtagmk1), frekvenci mikrokontroléru (14745600Hz) a jako poslední, port v počítači ve kterém máme zapojený programátor.

Pomocí příkazu DDR (Data Direction Register) nastavujeme pro námi zvolených 8 bitů, zda bity jsou vstup či výstup

Schéma zapojení



Popis Programu

Světelný řádek jsme si rozdělili na 6 segmentů, každý segment obsahuje 8x8 LED diod pro zobrazované číslice, ke každému segmentu jsme přiřadili int číslic (hodiny1, hodiny2, minuty1..., sekundy2).

Samotný námi vytvořený font číslic je uložený v metodě „cislice“. Každá číslice obsahuje osm 8-bitových hodnot pro 8 sloupců, kde zapisujeme log. 1 a log.0 podle toho kde chceme aby se nám LED dioda rozsvítila (log. 0 rozsvíceno, log. 1 zhasnuto)

Dále jsme si vytvořili „int sloupce“, který funguje jako zapínání námi určeného sloupce při zapisování. Sloupce zapínáme registrem C7 log. 1 stejně jako každý jiný registr. Po zapnutí n sloupce se na světelném řádku zobrazí data na každem n sloupci každého segmentu podle toho jaká data máme uložena v registrech C0-C5

Před zapnutím programu musíme nastavit DDRB a DDRD na samé log. 1, tím nastavíme naše používané bity jako výstup aby jsme na ně mohli zapisovat. Registry (PORTB) nastavíme na log. 0, jelikož se zapínají v log. 1 a to samé uděláme pro data (PORTD) akorát nastavíme log. 1, to celé je v metodě „nastaveni“

Zapisování číslic funguje následovně, na PORTD si připravíme data která chceme zobrazit a zapneme registr (C0-C5) na kterém data zobrazíme, vytvoříme zpoždění na 1µs aby se data bez problémů vložila do registru a registr vypneme, tím data uložíme v registru a na PORTD zapíšeme opět samé log. 1, jako finále zapneme registr C7 a určitý sloupec, opět vytvoříme zpoždění a registr vypneme a uložíme do něj data.

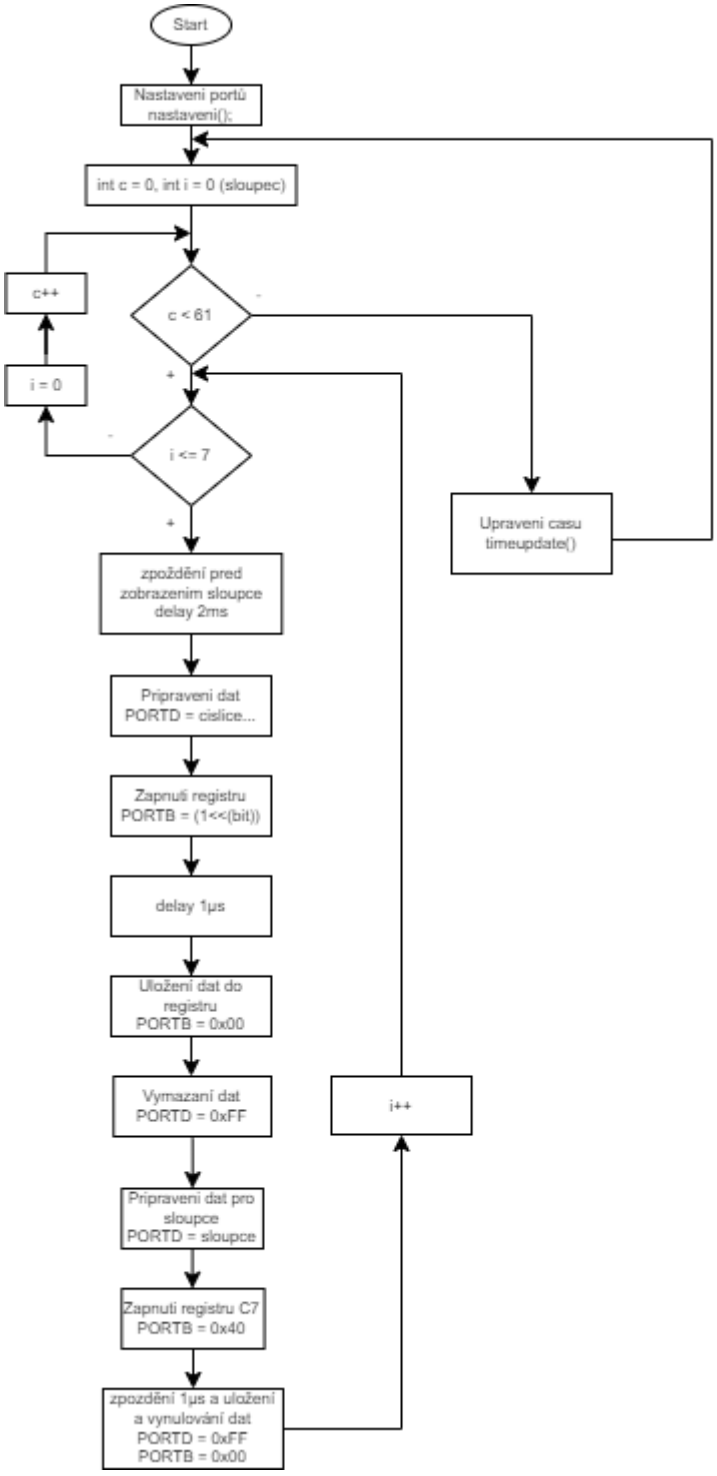
Při spuštění programu se první vykonná nastavení potom se přejde do cyklu while(1) a spustí se cyklus, který 61x zapíše všechny číslice na všech segmentech, po vykonání tohoto cyklu uběhne zhruba 0,995s. Spustí se metoda „timeupdate“ kdy se přičte sekunda (sekunda2++) a zpracují se inty hodiny1, hodiny2..., sekundy2 tak aby se chovali jako hodiny (když sekundy2 je 10, přičte se sekundy1 a vynuluje sekundy2... a takhle až do 24 hodin)

Popis proměnných

Typ	Proměnná	Účel
int	**cislice**	Uložení fontu číslic
int	**sloupce**	Zapínání sloupců
int	**hodiny1**	Segment pro 1. číslici hodin
int	**hodiny2**	Segment pro 2. číslici hodin
int	**minuty1**	Segment pro 1. číslici minut
int	**minuty2**	Segment pro 2. číslici minut
int	**sekundy1**	Segment pro 1. číslici sekund
int	**sekundy2**	Segment pro 2. číslici sekund

Typ	Argumenty	Metoda	Účel
void		**timeupdate**	Zpracování intů pro segmenty
void		**nastaveni**	Připravení PORTB a PORTD
void	int segment, int cislo, int bit	**zapsanicislice**	Zapsani cislice do segmentu
void	int cislo	**zapnutisloupcu**	Zapnuti sloupcu pro segmenty

Vývojový diagram



Výpis programu:

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>

int cislice[10][8] =
{
    {0xFF, 0x00, 0x00, 0x3C, 0x3C, 0x00, 0x00, 0xFF},//0
    {0xFF, 0xE3, 0xC7, 0x8F, 0x00, 0x00, 0x00, 0xFF},//1
    {0xFF, 0x20, 0x20, 0x24, 0x24, 0x04, 0x04, 0xFF},//2
    {0xFF, 0x3C, 0x3C, 0x24, 0x24, 0x00, 0x00, 0xFF},//3
    {0xFF, 0x07, 0x07, 0xE7, 0xC0, 0xC0, 0xE7, 0xFF},//4
    {0xFF, 0x04, 0x04, 0x24, 0x24, 0x20, 0x20, 0xFF},//5
    {0xFF, 0x00, 0x00, 0x24, 0x24, 0x20, 0x20, 0xFF},//6
    {0xFF, 0x3F, 0x3F, 0x33, 0x00, 0x00, 0xF3, 0xFF},//7
    {0xFF, 0x00, 0x00, 0x24, 0x24, 0x00, 0x00, 0xFF},//8
    {0xFF, 0x04, 0x04, 0x24, 0x24, 0x00, 0x00, 0xFF},//9
};

int sloupce[8] = {0x7F, 0xBF, 0xDF, 0xEF, 0xF7, 0xFB, 0xFD, 0xFE};

int hodiny1 = 0;
int hodiny2 = 0;
int minuty1 = 0;
int minuty2 = 0;
int sekundy1 = 0;
int sekundy2 = 0;

void timeupdate() {
    sekundy2++;
    if (sekundy2 == 10) {
        sekundy2 = 0;
        sekundy1++;
    }
    if (sekundy1 == 6) {
        sekundy1 = 0;
        minuta2++; }
    if (minuta2 == 10) {
        minuta2 = 0;
        minuta1++; }
    if (minuta1 == 6) {
        minuta1 = 0;
        hodiny2++; }
    if (hodiny2 == 10) {
        hodiny2 = 0;
        hodiny1++; }
    if (hodiny1 == 2 && hodiny2 == 4){
        hodiny1 = 0;
        hodiny2 = 0;
        minuta1 = 0;
        minuta2 = 0;
        sekundy1 = 0;
        sekundy2 = 0;
    }

}

void nastaveni(void){
    DDRB = 0xFF; //VYSTUP
    DDRD = 0xFF; //VYSTUP
    PORTB = 0x00; //REGISTRY, ZAPINAJI SE V LOG. 1
    PORTD = 0xFF; //DATA, ZAPINAJI SE V LOG. 0
}

void zapsanicislice(int segment, int cislo, int bit) {
    PORTD = cislice[segment][cislo]; //PRIPRAVENI DAT
    PORTB=(1<<(bit)); //ZAPNUTI REGISTRU PRO URCITY "SEGMENT"
    _delay_us(1);
    PORTB=0x00; //ULOZENI DAT DO REGISTU
```

```
    PORTD=0xFF; //VYNULOVANI DAT
}

void zapnutisloupcu(int cislo){
    PORTD = sloupce[cislo]; //PRIPRAVENI DAT PRO ZAPNUTI SLOUPCU
    PORTB=0x40; //ZAPNUTI C7
    _delay_us(1);
    PORTB=0x00; //ULOZENI DAT DO REGISTRU
    PORTD=0XFF; //VYNULOVANI DAT
}

int main(void){

    nastaveni();

    while (1) {
        for (int c = 0; c <= 61; c++) {
            for (int i = 0; i <= 7; i++) {

            _delay_ms(2);
            zapsanicislice(hodiny1, i, 0);
            zapsanicislice(hodiny2, i, 1);
            zapsanicislice(minuty1, i, 2);
            zapsanicislice(minuty2, i, 3);
            zapsanicislice(sekundy1, i, 4);
            zapsanicislice(sekundy2, i, 5);
            zapnutisloupcu(i);
            }
        }
        timeupdate();
    }
        return 0;
    }
```

Odpovědi na otázky

1. Pro stavbu větších celků je možné s výhodou použít tzv. „smart pixels“. Popište princip této technologie a naznačte jak by vypadalo použití ve světelném řádku podobném tomu z úlohy.
   - Smart pixels znamená, že můžeme kontrolovat barvu každé LED diody kterou používáme. Každá LED dioda by se skládala z dalších 3 LED diod která umí svítit v 3 barvách, zelená, modrá a červena a podle intenzity svítení každé barvy ve finále dokážeme vytvořit jakoukoliv barvu kterou potřebujeme na každé LED diodě. Takováhle dioda se nazývá „pixel“ (
   - Tyto pixely se pak používají v LED pásku, to by znamenalo, že náš světelný řádek by byl jako jeden celý LED pásek a nemuselo by se zobrazovat furt dokola každá číslice ale stačila by zapsat pouze jednou, ale velkou nevýhodou by bylo, že by jsme používali trojnásobek LED diod (1*)
2. V případě požadavku na podporu přímého zobrazení např. video signálu by bylo potřeba implementovat vhodné vstupní rozhraní. Pokuste se popsat kódování obrazu v rámci analogového rozhraní VGA.
3. Při použití multiplexního řízení LED je nutné pamatovat na proud tekoucí čipem LED s ohledem na její svítivost. Jak bude tento proud vypadat při použití 8 LED tak, aby jejich svítivost byla srovnatelná se statickým (např. paralelním) ovládáním bez multiplexu? Jakým parametrem LED budete omezeni?

Závěr

S minimálními detaily úloha vyšla přesně. Přesně by vyšla kdybychom mohli nějakým způsobem udělat cyklus místo 61x zapsání, na 61,2x +- což nelze. Také by šlo nastavit v procesoru pomocí Timeru1 nebo Timeru2 delay přesně na 1 sekundu, což je ta těžší cesta, ale pro naše potřeby tato přesnost bohatě stačí.[1]

[1] https://www1.lightorama.com/smart-pixels/