Středoškolská technika 2012

Setkání a prezentace prací středoškolských studentů na ČVUT

INFORMAČNÍ LED DISPLEJ

Martin Uhlík

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

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Informační LED displej

Martin Uhlík

Ostrava 2012

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor: 10. Elektrotechnika, elektronika a telekomunikace

Informační LED displej

Autor: Martin Uhlík

Škola: Střední průmyslová škola elektrotechniky a informatiky, Kratochvílova 7, Ostrava

Konzultant: Ing. Ladislav Škapa

Ostrava 2012
Moravskoslezský kraj

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně, použil jsem pouze podklady (literaturu, SW atd.) citované v práci a uvedené v přiloženém seznamu a postup při zpracování práce 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 …….. Podpis: …….

ANOTACE

Obsahem práce je návrh konstrukce panelového LED displeje, který je možno ovládat po sériové lince jednoduchým protokolem tak, aby byl displej použitelný i bez počítače PC. V budoucnu bude možné připojit malý terminál, který bude obsahovat pomocný malý displej, aby jej bylo možno umístit mimo dohled na LED panel, a pro zadávání znaků bude mít k dispozici port PS/2 pro běžnou počítačovou klávesnici.

Panelový displej LED je řešen modulově tak, aby se jeho délka dala upravit dle konečného použití a umístění. Délka displeje je omezena pouze velikostí paměti. v mikrokontroléru, nyní je maximum 7 modulů (výsledná délka displeje cca 80cm).

Displej je použitelný na různá místa, kde je třeba informovat větší množství lidí.

Klíčová slova: Digitální, displej, LED, běžící text

Obsah

1. Úvod

K realizaci velkého displeje je neprve třeba zvážit všechny možné řešení a z nich vybrat nejvhodnější pro realizaci k danému účelu. Jelikož každé řešení má i své úskalí, je nutno promyslet i tato úskalí.

Jako ideální řešení se na první pohled jeví displej na jednom kusu desky plošného spoje (DPS), který by byl i částečně samonosný. Jenže vyrábět DPS takových rozměrů je problematické a to nejen kvůli ceně, ale kvůli samotné fyzické realizaci předlohy.

Další možnost je displej vyrobit modulově tak, že na každém modulu bude určitý počet led (nejlépe mocnina 2). Jelikož tato varianta vychází nejlépe v poměru cena/výkon, byla zvolena tato varianta.

Co se týče mechanické konstrukce, displej je možno umístit buď do krabice tak, aby byly vidět pouze LE diody, nebo je možno displej udělat samonosný, aby se samotná DPS dala připevnit na pevný podklad. Po zvážení jednotlivých aspektů a vzhledu jednotlivých variant byla zvolena varianta umístění displeje do celoprůhledné krabice z plexiskla zepředu orámované hliníkovými profily.

2. Panelový LED displej

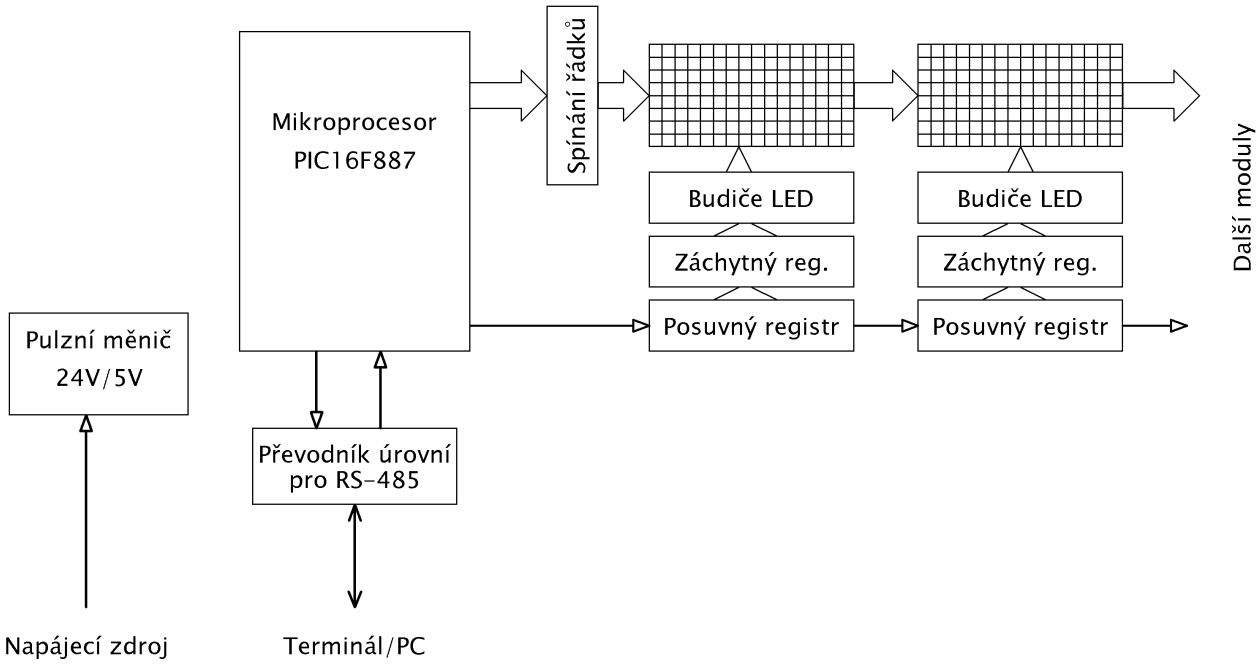
2.1. Princip funkce

Samotné principální schéma displeje je poměrně jednoduché. LED diody jsou zapojeny v matici po jednotlivých modulech. V jednom modulu je 8 řádků a 16 sloupců LED diod. Multiplexní ovládání je řešeno spínáním po řádcích, aby byla doba svitu jednoho řádku co největší. Pokud by bylo multiplexní ovládání řešeno spínáním po sloupcích, výsledná doba svitu jednoho sloupce by byla příliš krátká a svit LED by byl nízký i přes velký proud. Díky použití záchytných registrů je možné do posuvných registrů nahrávat data pro následující řádek, zatímco stávající svítí. Tím se minimalizuje doba „tmy“, která by byla nutná pro nahrávání dat bez použití záchytných registrů.

Jelikož displej bude pravděpodobně vzdálen od terminálu více než 1m, je nutno použít pro asynchronní komunikaci převodník úrovní. Jako vhodné řešení jsem zvolil sběrnici na principu RS-422. Tato sběrnice má pro každý směr dat dva vodiče a je řešena jako rozdílová, aby vlivem cizího rušení nedošlo ke špatnému vyhodnocení dat.

Napájecí napětí 24V bylo zvoleno záměrně, aby nedocházelo k proudovému namáhání kontaktů a vodičů. Pokud totiž uvažujeme např. s 6 moduly, proudem jednou LED 20mA a 16 LED v jednom modulu, nejvyšší v jednu chvíli odebíraný proud je $I_{\max} = I_{LED} \cdot N = 20mA \cdot (6 \cdot 16) = 1920mA$. Při uvažované účinnosti spínaného měniče n = 80% vychází vstupní proud při 24V pouze 480mA, což představuje úsporu na mědi vodičů a umožňuje použití menších napájecích konektorů.

Principální blokové schéma je uvedeno na následujícím obrázku:



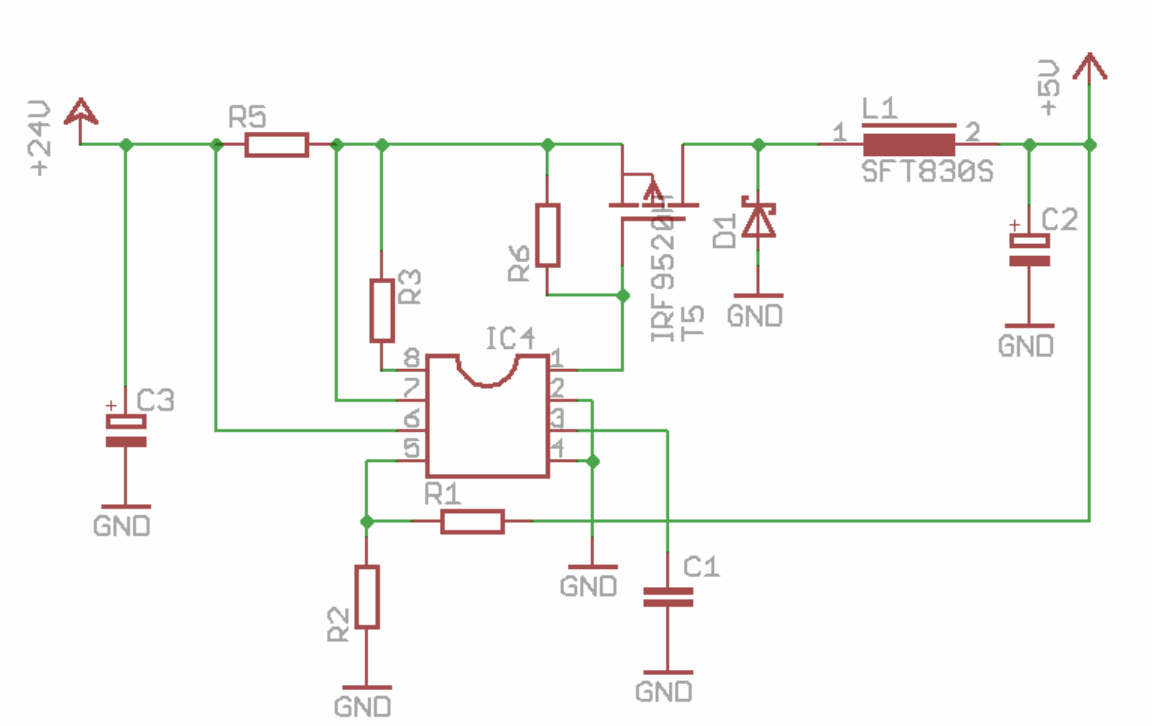
Obr. 1 – blokové schéma LED displeje

2.2. Důležitá místa zapojení LED displeje

Zapojení vychází ze základních principů multiplexního displeje a co se týče samotné displejové části, je velmi jednoduché díky použitým součástkám.

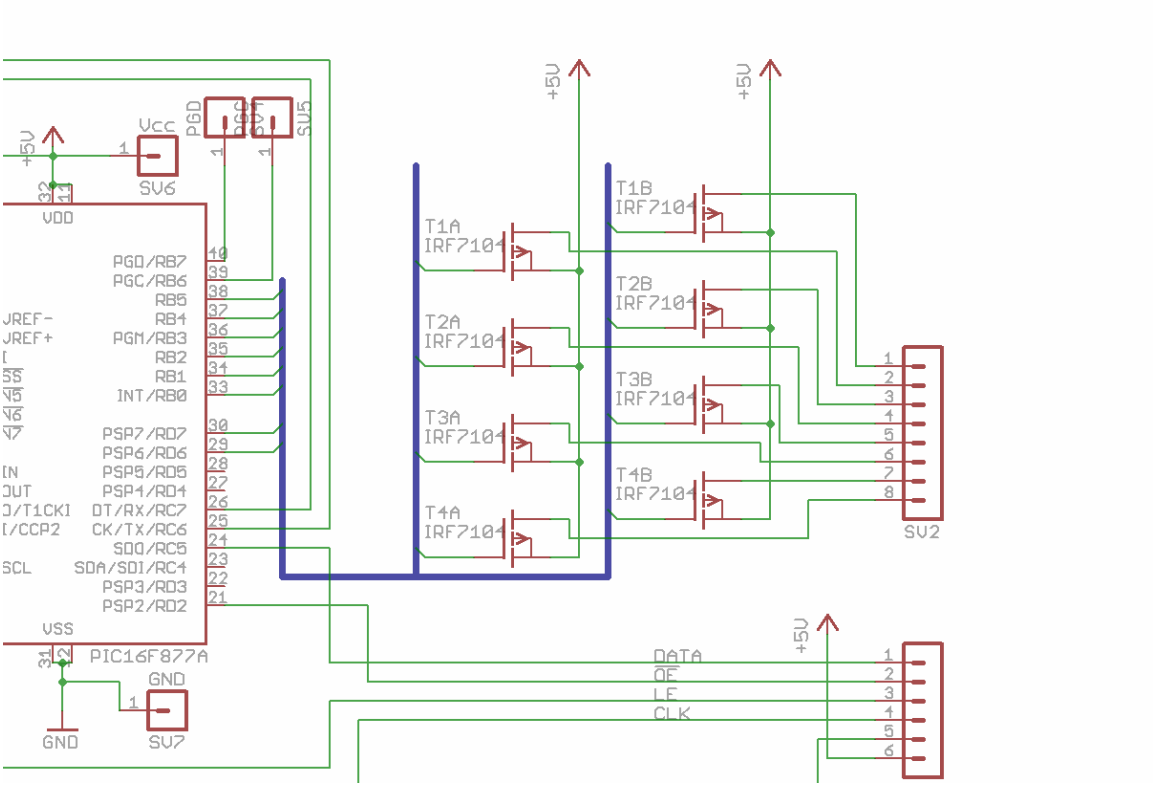
Napájecí zdroj

Spínaný DC/DC měnič z 24V na 5V je řešen katalogovým zapojením obvodu MC34063 s pomocným spínacím MOSFET tranzistorem.



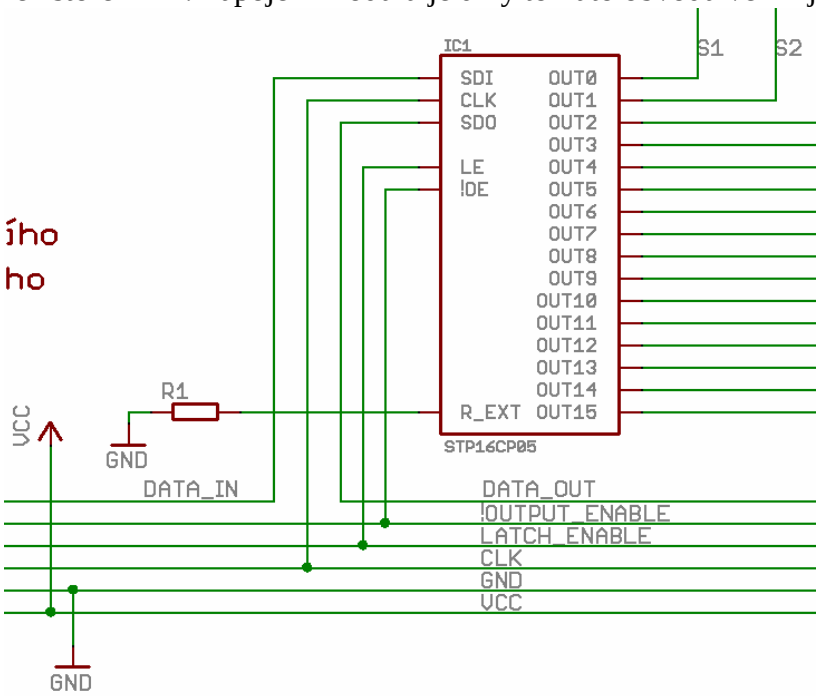
Buzení řádků

Jednotlivé řádky multiplexu jsou spínány tranzistory MOSFET s kanálem P. Tranzistory mají v sepnutém stavu tak malý odpor, že se zahřívají jen nepatrně oproti bipolárním tranzistorům.



Zapojení LED modulu

Pro ovládání LED matice jsem vybral obvod STP16CP05, který obsahuje všechny tři obvody potřebné pro buzení sloupců – posuvný registr (sériově-paralelní), záchytný registr a budič LE diod s konstantním proudem. Proud diodami lze nastavit vnějším rezistorem R1. Zapojení modulu je díky tomuto obvodu velmi jednoduché.

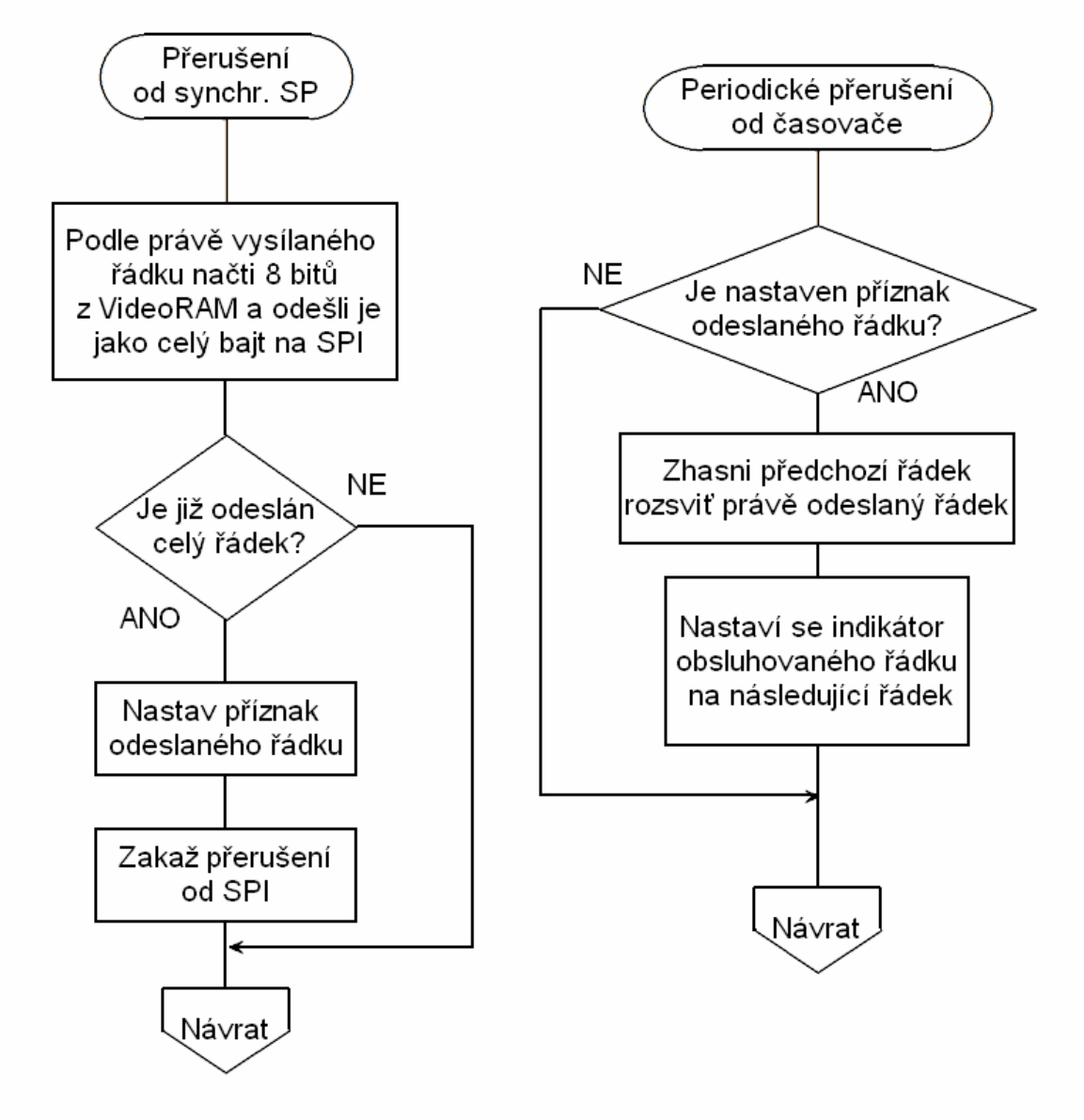


2.3. Důležitá místa v programu řídícího modulu

Program je navržen jako dva navzájem nezávislé funkční celky. Jeden vybírá data z tzv. VideoRAM, přepočítává je pro sériový port SPI a obsluhuje multiplex displeje. Druhý získává data z pevné paměti na základě zásobníku s uloženým textem a tyto data vkládá do VideoRAM. Paměť VideoRAM tedy obsahuje obraz displeje v paměti mikrokontroléru. Na displeji tedy lze zobrazit libovolný útvar, obrázek či text.

V rámci druhého celku, který se stará o vkládání znaků do VideoRAM funguje i podprogram, který má na starosti příjem znaků z asynchronního portu. Protokol je popsán níže.

Principální vývojový diagram programu na obsluhu multiplexního displeje:



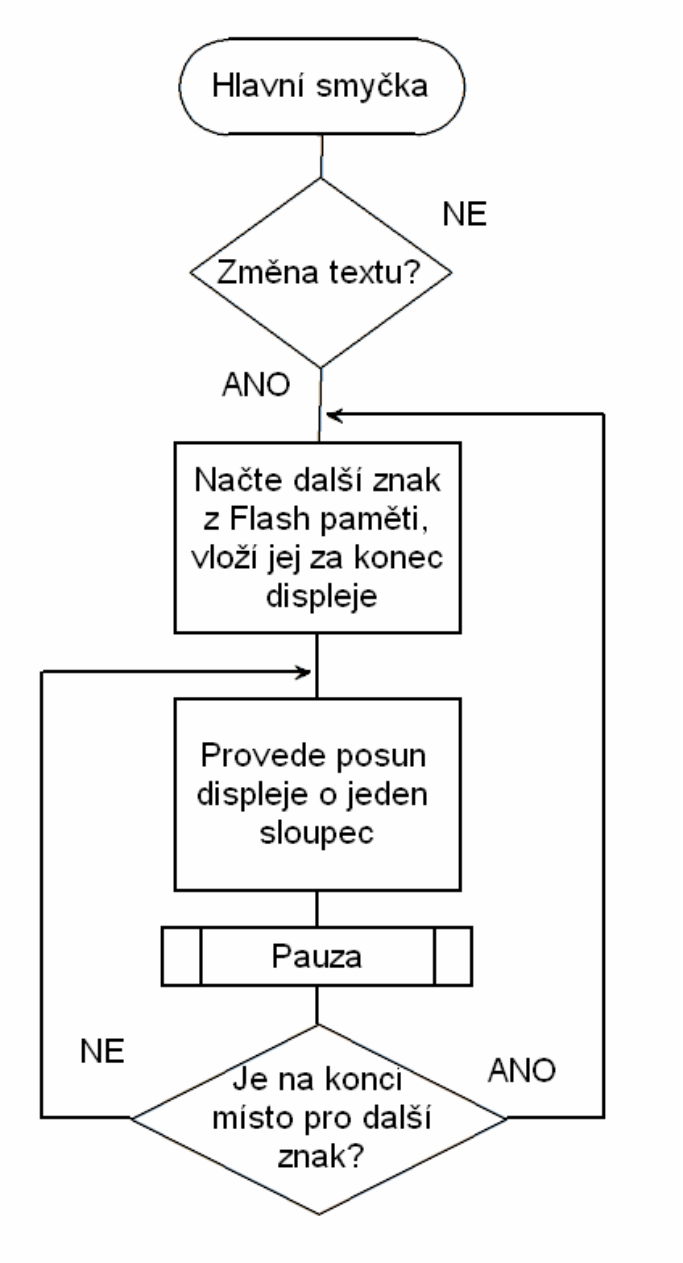
Přepočet dat je nutný, jelikož v paměti mikroprocesoru je obraz uložen po sloupcích (jeden byte je jeden sloupec displeje), ale posuvné registry vyžadují data řádku.
Z každého bajtu VideoRAM je tedy třeba načíst bit pro příslušný řádek a vložit jej bitovou rotací do pomocného registru. Po načtení 8 bitů se tento bajt odešle do posuvných registrů. Tento proces se opakuje, až se odešlou bity pro celý řádek.

Podprogram přerušení od sériového portu zpracovává protokol a podle příkazu a příchozích dat je ukládá to příslušných registrů. Text k zobrazení je ukládán do

zásobníku, z nějž poté program v hlavní smyčce čte a vkládá znaky k zobrazení do VideoRAM.

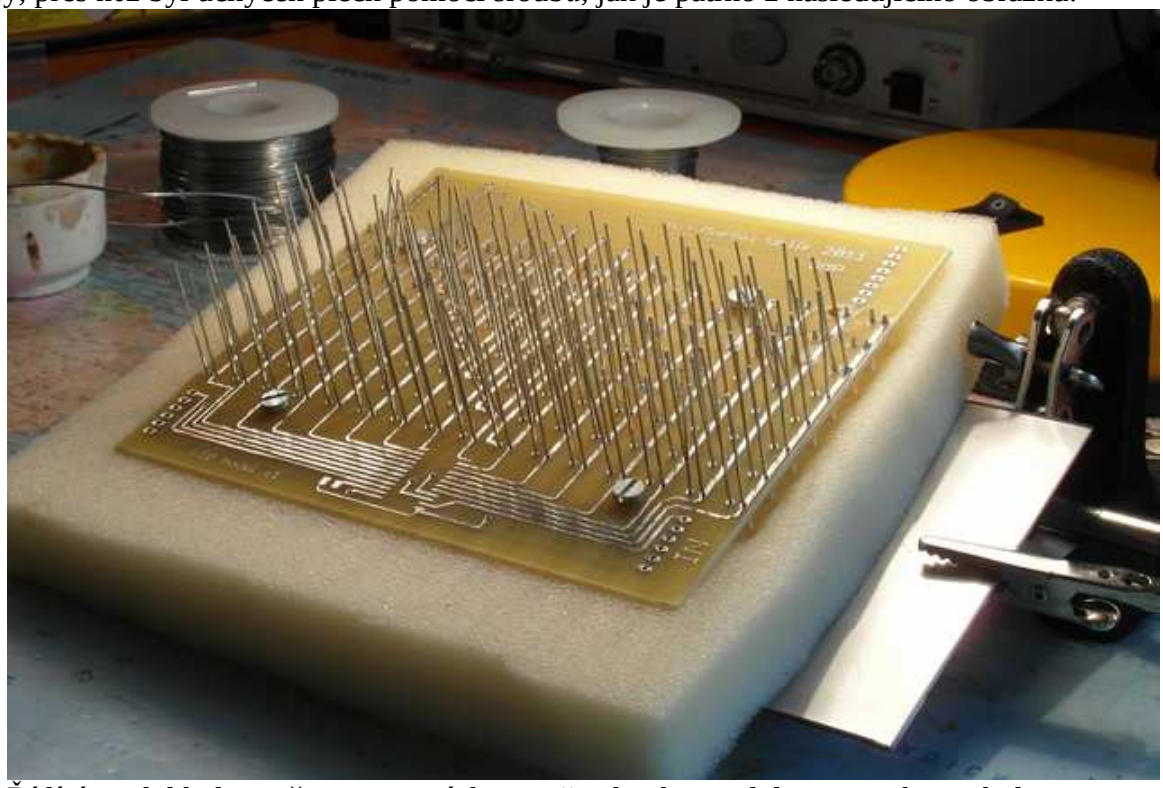
Program v hlavní smyčce kontroluje informační příznaky z přerušení, načítá text ze zásobníku a podle něj načítá znaky z Flash paměti a vkládá je do paměti k zobrazení na displeji. Posun textu funguje tak, že se vždy posune obsah všech sloupců o jedno místo. VideoRAM je delší než skutečná zobrazovaná plocha a tudíž je na jejím konci „neviditelná oblast“. Jakmile se v této oblasti uvolní místo pro další znak, je načten z paměti Flash a vložen do tohoto volného místa. Rychlost posuvu je určena registrem, který je možno pomocí příkazu po sériovém portu nastavit na určitou hodnotu. Minimální hodnota rolování zastaví, maximální jej učiní nečitelný.

Principální zjednodušené schéma programu hlavní smyčky je na obrázku zde:



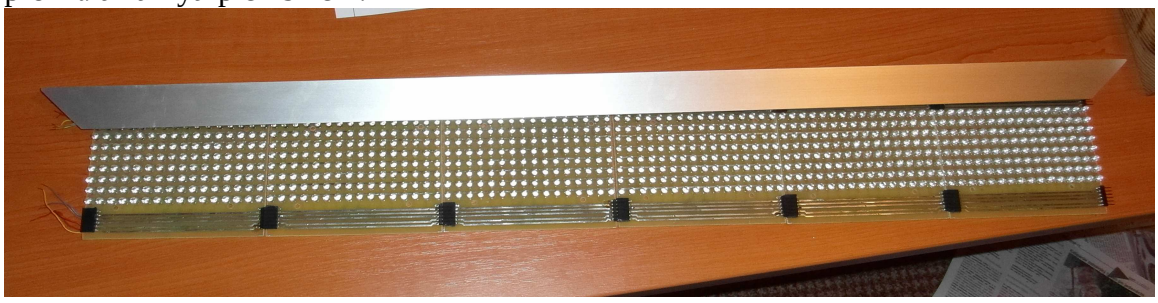
2.4. Návrh konstrukce a výroba

Jak již bylo uvedeno výše, samotný displej je řešen modulově. Každý modul ma svou vlastní desku plošného spoje, na které je umístěno 8*16 LED a jeden obvod pro jejich řízení. Pro nejlepší vzhled a spolehlivou a méně pracnou výrobu je vhodné, aby tato deska byla oboustranná a prokovená. V případě neprokovené desky je nutno zapájet diody z obou stran a to by vedlo k různému úhlu mezi jednotlivými diodami, nerovnostem a celkový vzhled desky by nebyl dobrý. V případě prokovené desky lze součástky zapájet pouze z jedné strany, což je velká výhoda a ušetření značného množství času. Veškeré výkresy DPS jsou obsaženy v příloze a to jak ve formátu pdf, tak v originálním formátu z návrhového systému Eagle. Osazování takového množství LED se neobejde bez přípravku na jejich vhodné uchycení. Do desky byly vyvrtány čtyři otvory, přes něž byl uchycen plech pomocí šroubů, jak je patrno z následujícího obrázku:



Řídící modul byl navržen ve stejných rozměrech jako modul s LED, aby jej bylo možno umístit do řady s LED moduly.

Po osazení všech desek následovalo jejich upevnění do řady pomocí hliníkových profilů a zakrytí plexisklem.



3. Komunikační protokol displeje

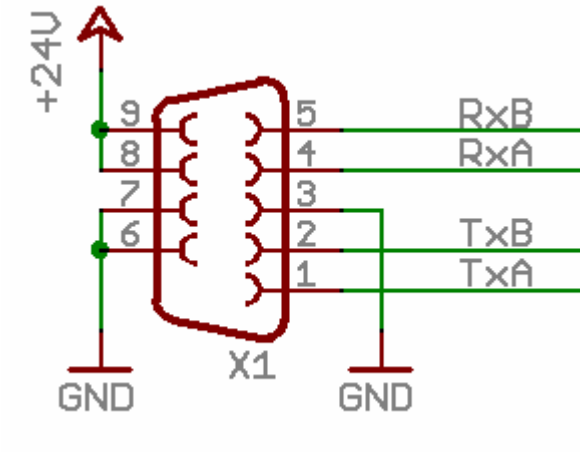
Protokol je navržen tak, aby byl jednoduše programově realizovatelný. Komunikace probíhá tzv. packtem. Jeden packet nemá omezenou délku, je uvozen Start a End bytem. Start byte může nabývat různých hodnot a slouží jako informace o druhu paketu. Zatím jsou k dispozici čtyři příkazy:

Text k zobrazení	01h
Změna rychlosti rotace	02h
Zastavení rotace	04h
Smazat displej	08h

Není vyloučeno, že se v budoucnu počet příkazů zvětší. Pro základní ovládání však stačí tyto čtyři.

Po příkazu text k zobrazení následuje samotný text, jehož délka je omezena pouze velikostí vnitřního zásobníku v mikroprocesoru, tedy velikostí paměti. Po vyplnění paměti velkou VideoRAM na text zbylo 160 bajtů, tedy 160 znaků jednoho nápisu. Ačkoliv to může vypadat jako málo, dá se toto obejít použitím vnějšího terminálu a zobrazováním více nápisů, kdy každý bude mít 160 znaků. Také je možné tyto nápisy uložit do EEPROM nebo Flash paměti mikrokontroléru a z ní je poté načítat.

Zapojení konektoru Cannon 9 je na následujícím obrázku a popis pinů je pro přehlednost uveden v tabulce:



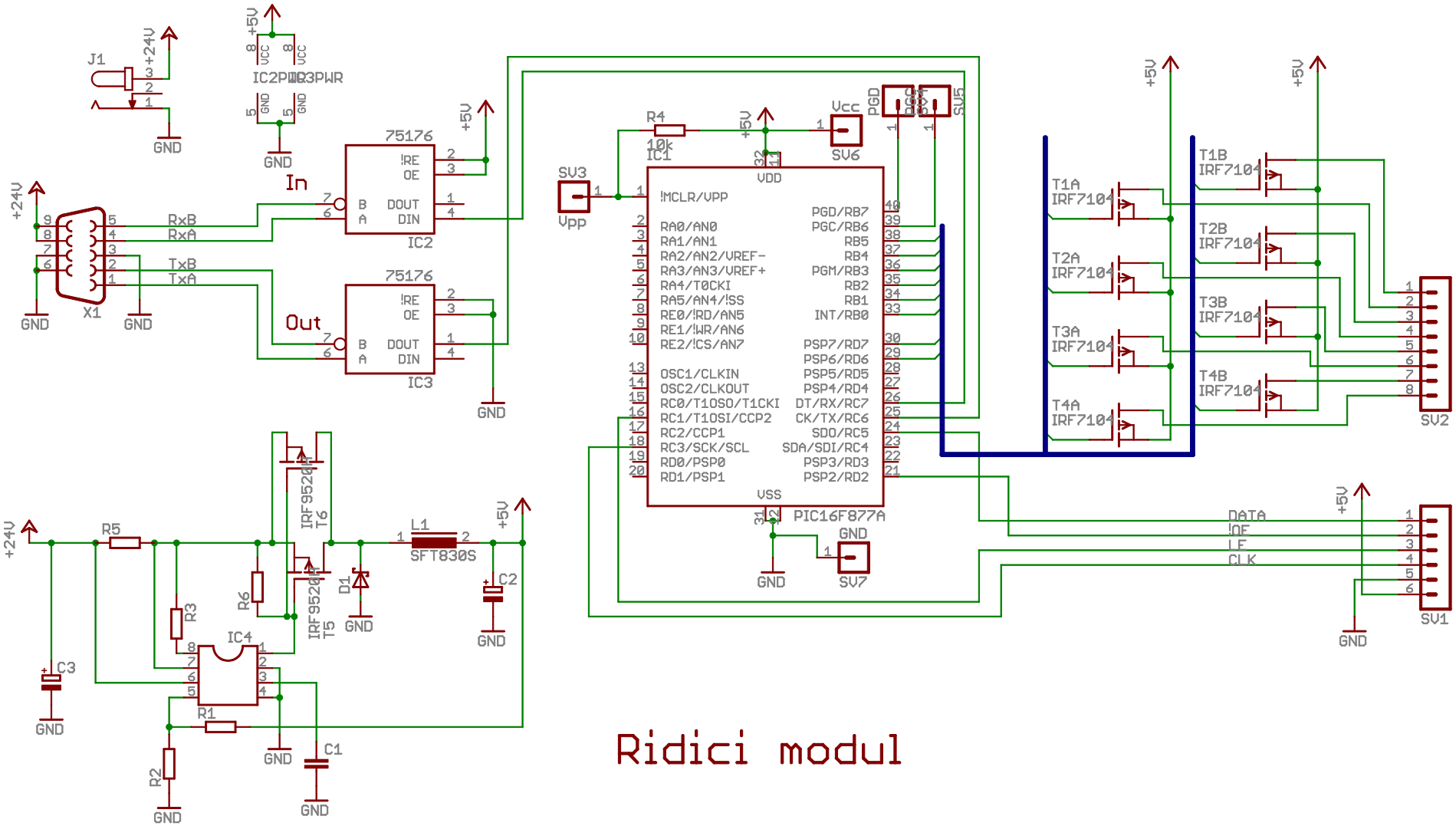
Pin	Funkce
1	Tx A
2	Tx B
4	Rx A
5	Rx B
3,6,7	Zem
8,9	+24V

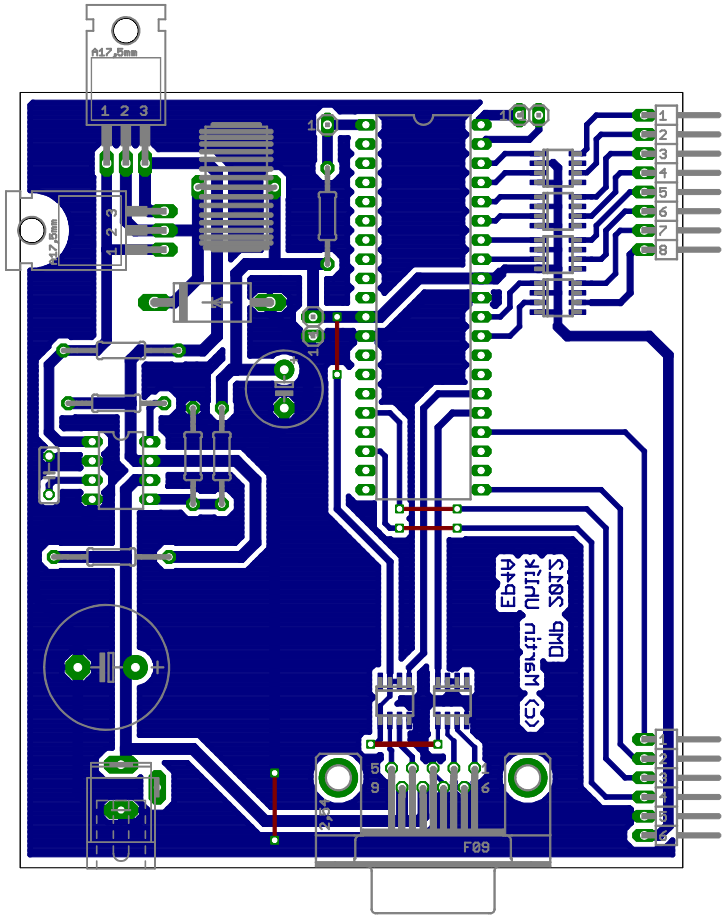
4. Závěr

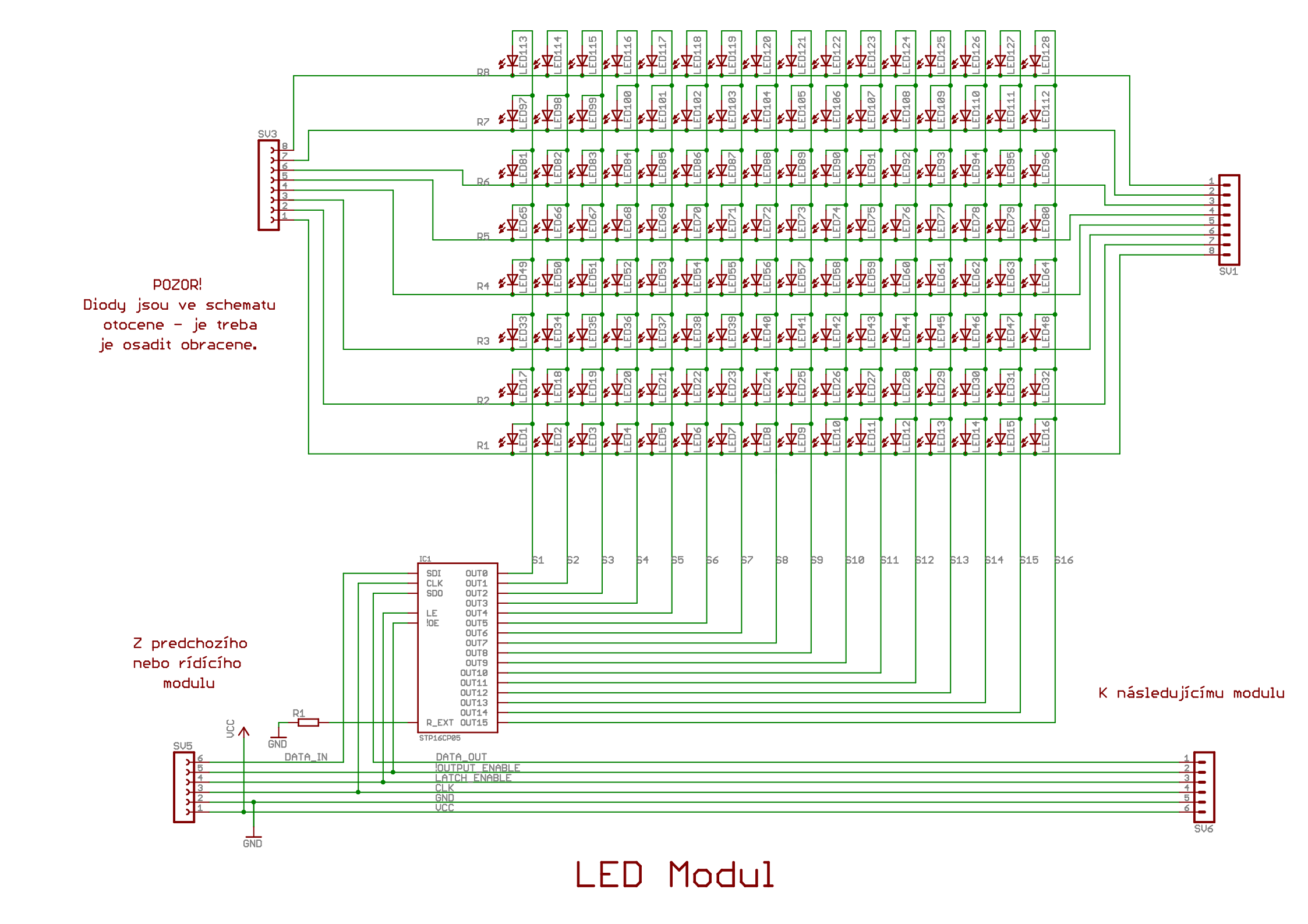
Displej je díky použití vysocesvítivých diod viditelný z dostatečné vzdálenosti. Ovládání je jednoduché a v případě doplnění terminálem k PS/2 klávesnici může být zcela nezávislý na počítači. Rozměry displeje jsou přijatelné pro zobrazení i dlouhého slova a proto bude čtení pohodlné, v případě, že se nastaví vhodná rychlost běžícího textu.

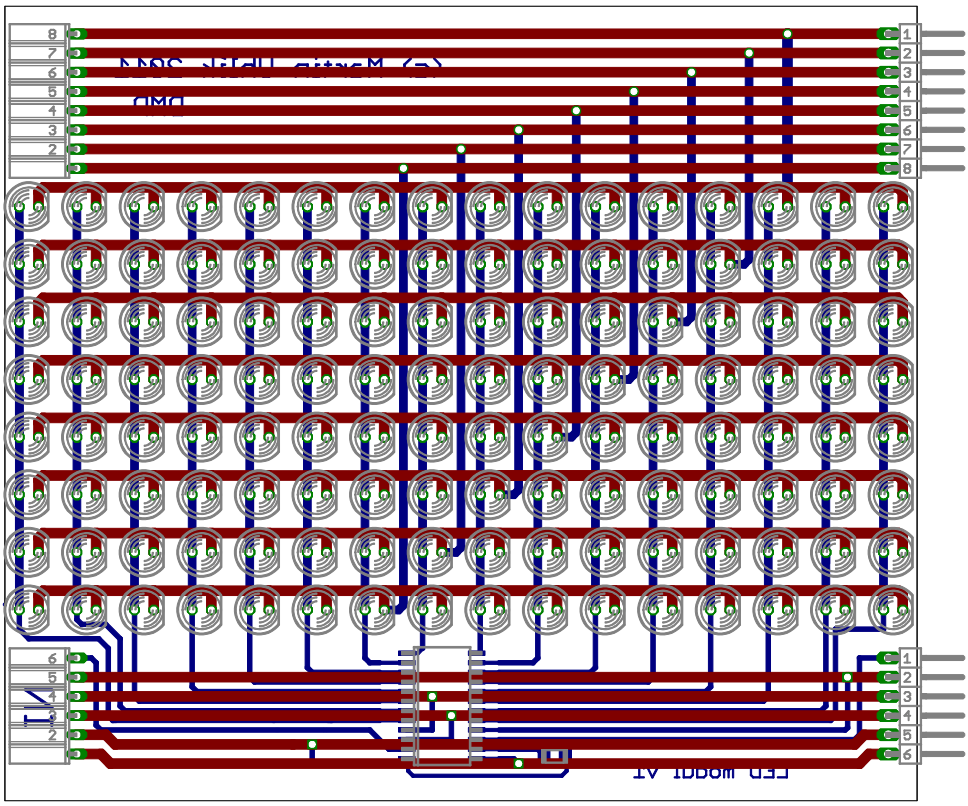
5. Použitá literatura

[1] http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00126634.pdf
[2] http://ww1.microchip.com/downloads/en/DeviceDoc/41291F.pdf


Ridici modul
PIC16F877A
75176
IC1
IC2
IC3
IC4
X1
J1
SV1
SV2
SV3
SV5
SV6
SV7
T1A
T1B
T2A
T2B
T3A
T3B
T4A
T4B
IRF7104
IRF9520
SFT830S
In
Out
RxB
RxA
TxB
TxA
DATA
!OE
LE
CLK
Vpp
Vcc
PGD
PGC
+5V
+24V
GND


1 2 3
F09


POZOR!
Diody jsou ve schematu otocene - je treba je osadit obracene.
Z predchozího nebo rídícího modulu
K následujícímu modulu
SV3
SV1
SV5
SV6
R1
R2
R3
R4
R5
R6
R7
R8
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16
IC1
SDI
CLK
SDO
LE
!OE
R_EXT
OUT0
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
OUT8
OUT9
OUT10
OUT11
OUT12
OUT13
OUT14
OUT15
STP16CP05
VCC
GND
DATA_IN
DATA_OUT
!OUTPUT_ENABLE
LATCH_ENABLE
CLK
GND
VCC
LED Modul



```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;                            Dlouhodobá maturitní práce 2012
                                ;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;    Název: Maticový LED displej - běžící text
                                        ;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;    Autor: Martin Uhlík, EP4A
                                                        ;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;    8. ledna 2012 - počátek psaní programu
                                                ;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;    11. ledna 2012 - okomentované důležité místa programu
                                ;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;    15. února 2012 - první zkušební test programu - nespolehlivé
                        ;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;    27. února 2012 - nový algoritmus obsluhy přerušení přepsán a
okomentován     ;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;    5. března 2012 - odlaďování nového algoritmu, hledání chyb
                        ;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;

        LIST    P = PIC16F887, R = DEC
        #include "p16f887.inc"
        errorlevel      -302, -202

;       org     2007h
;       data    20E2h
;       org     2008h
;       data    3fffh

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Definice portů na něž jsou připojeny budiče řádků displeje
;;; Budiče jsou provedeny tranzistory PNP, výstupy jsou tedy aktivní v nule
#define R0      PORTA,0
#define R1      PORTA,1
#define R2      PORTB,5
#define R3      PORTB,4
#define R4      PORTB,3
#define R5      PORTB,2
#define R6      PORTB,1
#define R7      PORTB,0

#define OE      PORTD,3
#define LE      PORTD,2

VideoRAM1       equ     110h    ;umístění 1. sektoru VideoRAM
konecVRAM1      equ     16Fh

VideoRAM2       equ     190h    ;umístění 2. sektoru VideoRAM
konecVRAM2      equ     19Fh

#define NaZacatek       StatProg,2      ;příznak informující program, že se skáče na začátek
VideoRAM (vždy po odeslání celého řádku)
#define Odeslano        StatProg,0      ;příznak pro přerušení od časovače, že může provést změnu
řádku
#define Sektor          StatProg,1      ;příznak ukazující, který sektor VideoRAM je právě zvolen



;blok paměti přístupné ve všech bankách

TmpW    equ     71h     ;zálohovací registry pro přerušení
TmpStat equ     72h
TmpFSR  equ     73h
TmpPCH  equ     74h
StatProg        equ     75h     ;registr pro různé příznaky
```

```
Radek   equ     20h             ;ukazuje číslo řádku, který právě zpracováváme
IntTmp0 equ     21h     ;pomocný registr pro přerušení
Bit     equ     22h     ;ukazuje počet zbývajících bitů k odeslání
Offset  equ     23h     ;offset v jednotlivém sektoru VideoRAM
TMP0    equ     24h     ;pomocný registr pro hlavní smyčku
TMP1    equ     25h
TMP2    equ     26h

        org     0
        goto    Start

        org     4
        movwf   TmpW            ;záloha registrů pro přerušení
        swapf   STATUS,w
        movwf   TmpStat
        movf    FSR,w
        movwf   TmpFSR
        movf    PCLATH,w
        movwf   TmpPCH
        clrf    PCLATH
        banksel 0
IntRozc
        btfsc   PIR1,SSPIF      ;nastalo-li přerušení od SPI, skočíme na jeho obsluhu
        goto    IntSPI

        btfsc   INTCON,T0IF     ;nastalo-li přerušení od Timeru0, skočíme na jeho obsluhu
        goto    IntTMR

ENDINT  movf    TmpPCH,w        ;obnovení reigstrů a návrat z přerušení
        movwf   PCLATH
        movf    TmpFSR,w
        movwf   FSR
        swapf   TmpStat,w
        movwf   STATUS
        swapf   TmpW,f
        swapf   TmpW,w
        retfie
;.....................
; Přerušení od sběrnice SPI, obsluhuje odesílání dat do posuvných registrů
IntSPI
        bcf     PIR1,SSPIF      ;vynuluje příznak přerušení a pomocný bajt
        clrf    IntTmp0
        movlw   .8              ;počet bitů na jednu várku odeslaných dat je osm
        movwf   Bit
        movf    SSPBUF,w        ;simuluje přečtení - vyprázdnění sériového portu kvůli správné
obsluze portu

        movf    Radek,w         ;podle právě zvoleného řádku skočí na odesílání dat pro příslušný
řádek
        addwf   PCL,f

        goto    Radek0
        goto    Radek1
        goto    Radek2
        goto    Radek3
        goto    Radek4
        goto    Radek5
        goto    Radek6
        goto    Radek7

Radek0
        movlw   VideoRAM1
        btfsc   Sektor          ;v případě, že Sektor==1, nastaví se počáteční adresa druhého
sektoru. Jinak se ponechá adresa prvního sektoru
        movlw   VideoRAM2
        movwf   FSR
        movf    Offset,w
        addwf   FSR,f           ;FSR (adresa počátku sektoru VRAM) + Offset -> FSR
        bcf     STATUS,C
        btfsc   INDF,0          ;nyní se do Carry flagu uloží hodnota konkrétního bitu z
příslušného sloupce
        bsf     STATUS,C
        rlf     IntTmp0,f       ;tento bit se "naroluje" do pomocného registru
        movf    Offset,f        ;kontrola, je-li offset již nula
        btfsc   STATUS,Z        ;jestli ano (Z==1) program pokračuje na změnu sektoru; jestli ne
(Z==0), program pokračuje v cyklu dále
```

```
        goto    NextSektor0
        decf    Offset,f        ;odečte od Offsetu jedničku
        goto    PokrRad0        ;a pokračuje na odečítání bitu
NextSektor0
        btfsc   Sektor          ;podle stavu bitu Sektor se zvolí následující sektor
        goto    NextSektor1_0   ;nyní je sektor dvě, změníme na jedničku
                                ;nebo je-li sektor v nule
        bsf     Sektor
        movlw   konecVRAM2
        movwf   Offset
PokrRad0
        decfsz  Bit,f           ;už byla odeslána osmice bitů?
        goto    Radek0          ;ještě ne, počítáme další bity
        goto    KonRad0
NextSektor1_0
        bcf     Sektor
        bsf     NaZacatek
        movlw   konecVRAM1
        movwf   Offset
KonRad0
        btfss   NaZacatek
        goto    BuffRad0
        bsf     Odeslano
        bcf     NaZacatek
        banksel PIE1
        bcf     PIE1,SSPIE
        banksel 0
BuffRad0
        movf    IntTmp0,w
        movwf   SSPBUF
        goto    ENDINT
;-------------------------------------------------------------------------------------------
Radek1
        movlw   VideoRAM1
        btfsc   Sektor          ;v případě, že Sektor==1, nastaví se počáteční adresa druhého
sektoru. Jinak se ponechá adresa prvního sektoru
        movlw   VideoRAM2
        movwf   FSR
        movf    Offset,w
        addwf   FSR,f           ;FSR (adresa počátku sektoru VRAM) + Offset -> FSR
        bcf     STATUS,C
        btfsc   INDF,1          ;nyní se do Carry flagu uloží hodnota konkrétního bitu z
příslušného sloupce
        bsf     STATUS,C
        rlf     IntTmp0,f       ;tento bit se "naroluje" do pomocného registru
        movf    Offset,f        ;kontrola, je-li offset již nula
        btfsc   STATUS,Z        ;jestli ano (Z==1) program pokračuje na změnu sektoru; jestli ne
(Z==0), program pokračuje v cyklu dále
        goto    NextSektor1
        decf    Offset,f        ;odečte od Offsetu jedničku
        goto    PokrRad1        ;a pokračuje na odečítání bitu
NextSektor1
        btfsc   Sektor          ;podle stavu bitu Sektor se zvolí následující sektor
        goto    NextSektor1_1   ;nyní je sektor dvě, změníme na jedničku
                                ;nebo je-li sektor v nule
        bsf     Sektor
        movlw   konecVRAM2
        movwf   Offset
PokrRad1
        decfsz  Bit,f           ;už byla odeslána osmice bitů?
        goto    Radek1          ;ještě ne, počítáme další bity
        goto    KonRad1
NextSektor1_1
        bcf     Sektor
        bsf     NaZacatek
        movlw   konecVRAM1
        movwf   Offset
KonRad1
        btfss   NaZacatek
        goto    BuffRad1
        bsf     Odeslano
        bcf     NaZacatek
        banksel PIE1
        bcf     PIE1,SSPIE
        banksel 0
BuffRad1
```

```
        movf    IntTmp0,w
        movwf   SSPBUF
        goto    ENDINT
;-------------------------------------------------------------------------------------------
Radek2
        movlw   VideoRAM1
        btfsc   Sektor          ;v případě, že Sektor==1, nastaví se počáteční adresa druhého
sektoru. Jinak se ponechá adresa prvního sektoru
        movlw   VideoRAM2
        movwf   FSR
        movf    Offset,w
        addwf   FSR,f           ;FSR (adresa počátku sektoru VRAM) + Offset -> FSR
        bcf     STATUS,C
        btfsc   INDF,2          ;nyní se do Carry flagu uloží hodnota konkrétního bitu z
příslušného sloupce
        bsf     STATUS,C
        rlf     IntTmp0,f       ;tento bit se "naroluje" do pomocného registru
        movf    Offset,f        ;kontrola, je-li offset již nula
        btfsc   STATUS,Z        ;jestli ano (Z==1) program pokračuje na změnu sektoru; jestli ne
(Z==0), program pokračuje v cyklu dále
        goto    NextSektor2
        decf    Offset,f        ;odečte od Offsetu jedničku
        goto    PokrRad2        ;a pokračuje na odečítání bitu
NextSektor2
        btfsc   Sektor          ;podle stavu bitu Sektor se zvolí následující sektor
        goto    NextSektor1_2   ;nyní je sektor dvě, změníme na jedničku
                                ;nebo je-li sektor v nule
        bsf     Sektor
        movlw   konecVRAM2
        movwf   Offset
PokrRad2
        decfsz  Bit,f           ;už byla odeslána osmice bitů?
        goto    Radek2          ;ještě ne, počítáme další bity
        goto    KonRad2
NextSektor1_2
        bcf     Sektor
        bsf     NaZacatek
        movlw   konecVRAM1
        movwf   Offset
KonRad2
        btfss   NaZacatek
        goto    BuffRad2
        bsf     Odeslano
        bcf     NaZacatek
        banksel PIE1
        bcf     PIE1,SSPIE
        banksel 0
BuffRad2
        movf    IntTmp0,w
        movwf   SSPBUF
        goto    ENDINT
;-------------------------------------------------------------------------------------------
Radek3
        movlw   VideoRAM1
        btfsc   Sektor          ;v případě, že Sektor==1, nastaví se počáteční adresa druhého
sektoru. Jinak se ponechá adresa prvního sektoru
        movlw   VideoRAM2
        movwf   FSR
        movf    Offset,w
        addwf   FSR,f           ;FSR (adresa počátku sektoru VRAM) + Offset -> FSR
        bcf     STATUS,C
        btfsc   INDF,3          ;nyní se do Carry flagu uloží hodnota konkrétního bitu z
příslušného sloupce
        bsf     STATUS,C
        rlf     IntTmp0,f       ;tento bit se "naroluje" do pomocného registru
        movf    Offset,f        ;kontrola, je-li offset již nula
        btfsc   STATUS,Z        ;jestli ano (Z==1) program pokračuje na změnu sektoru; jestli ne
(Z==0), program pokračuje v cyklu dále
        goto    NextSektor3
        decf    Offset,f        ;odečte od Offsetu jedničku
        goto    PokrRad3        ;a pokračuje na odečítání bitu
NextSektor3
        btfsc   Sektor          ;podle stavu bitu Sektor se zvolí následující sektor
        goto    NextSektor1_3   ;nyní je sektor dvě, změníme na jedničku
                                ;nebo je-li sektor v nule
        bsf     Sektor
```

```
        movlw   konecVRAM2
        movwf   Offset
PokrRad3
        decfsz  Bit,f           ;už byla odeslána osmice bitů?
        goto    Radek3          ;ještě ne, počítáme další bity
        goto    KonRad3
NextSektor1_3
        bcf     Sektor
        bsf     NaZacatek
        movlw   konecVRAM1
        movwf   Offset
KonRad3
        btfss   NaZacatek
        goto    BuffRad3
        bsf     Odeslano
        bcf     NaZacatek
        banksel PIE1
        bcf     PIE1,SSPIE
        banksel 0
BuffRad3
        movf    IntTmp0,w
        movwf   SSPBUF
        goto    ENDINT
;-------------------------------------------------------------------------------------------
Radek4
        movlw   VideoRAM1
        btfsc   Sektor          ;v případě, že Sektor==1, nastaví se počáteční adresa druhého
sektoru. Jinak se ponechá adresa prvního sektoru
        movlw   VideoRAM2
        movwf   FSR
        movf    Offset,w
        addwf   FSR,f           ;FSR (adresa počátku sektoru VRAM) + Offset -> FSR
        bcf     STATUS,C
        btfsc   INDF,4          ;nyní se do Carry flagu uloží hodnota konkrétního bitu z
příslušného sloupce
        bsf     STATUS,C
        rlf     IntTmp0,f       ;tento bit se "naroluje" do pomocného registru
        movf    Offset,f        ;kontrola, je-li offset již nula
        btfsc   STATUS,Z        ;jestli ano (Z==1) program pokračuje na změnu sektoru; jestli ne
(Z==0), program pokračuje v cyklu dále
        goto    NextSektor4
        decf    Offset,f        ;odečte od Offsetu jedničku
        goto    PokrRad4        ;a pokračuje na odečítání bitu
NextSektor4
        btfsc   Sektor          ;podle stavu bitu Sektor se zvolí následující sektor
        goto    NextSektor1_4   ;nyní je sektor dvě, změníme na jedničku
                                ;nebo je-li sektor v nule
        bsf     Sektor
        movlw   konecVRAM2
        movwf   Offset
PokrRad4
        decfsz  Bit,f           ;už byla odeslána osmice bitů?
        goto    Radek4          ;ještě ne, počítáme další bity
        goto    KonRad4
NextSektor1_4
        bcf     Sektor
        bsf     NaZacatek
        movlw   konecVRAM1
        movwf   Offset
KonRad4
        btfss   NaZacatek
        goto    BuffRad4
        bsf     Odeslano
        bcf     NaZacatek
        banksel PIE1
        bcf     PIE1,SSPIE
        banksel 0
BuffRad4
        movf    IntTmp0,w
        movwf   SSPBUF
        goto    ENDINT
;-------------------------------------------------------------------------------------------
Radek5
        movlw   VideoRAM1
        btfsc   Sektor          ;v případě, že Sektor==1, nastaví se počáteční adresa druhého
sektoru. Jinak se ponechá adresa prvního sektoru
```

```
        movlw   VideoRAM2
        movwf   FSR
        movf    Offset,w
        addwf   FSR,f           ;FSR (adresa počátku sektoru VRAM) + Offset -> FSR
        bcf     STATUS,C
        btfsc   INDF,5          ;nyní se do Carry flagu uloží hodnota konkrétního bitu z
příslušného sloupce
        bsf     STATUS,C
        rlf     IntTmp0,f       ;tento bit se "naroluje" do pomocného registru
        movf    Offset,f        ;kontrola, je-li offset již nula
        btfsc   STATUS,Z        ;jestli ano (Z==1) program pokračuje na změnu sektoru; jestli ne
(Z==0), program pokračuje v cyklu dále
        goto    NextSektor5
        decf    Offset,f        ;odečte od Offsetu jedničku
        goto    PokrRad5        ;a pokračuje na odečítání bitu
NextSektor5
        btfsc   Sektor          ;podle stavu bitu Sektor se zvolí následující sektor
        goto    NextSektor1_5   ;nyní je sektor dvě, změníme na jedničku
                                ;nebo je-li sektor v nule
        bsf     Sektor
        movlw   konecVRAM2
        movwf   Offset
PokrRad5
        decfsz  Bit,f           ;už byla odeslána osmice bitů?
        goto    Radek5          ;ještě ne, počítáme další bity
        goto    KonRad5
NextSektor1_5
        bcf     Sektor
        bsf     NaZacatek
        movlw   konecVRAM1
        movwf   Offset
KonRad5
        btfss   NaZacatek
        goto    BuffRad5
        bsf     Odeslano
        bcf     NaZacatek
        banksel PIE1
        bcf     PIE1,SSPIE
        banksel 0
BuffRad5
        movf    IntTmp0,w
        movwf   SSPBUF
        goto    ENDINT
;-------------------------------------------------------------------------------------------
Radek6
        movlw   VideoRAM1
        btfsc   Sektor          ;v případě, že Sektor==1, nastaví se počáteční adresa druhého
sektoru. Jinak se ponechá adresa prvního sektoru
        movlw   VideoRAM2
        movwf   FSR
        movf    Offset,w
        addwf   FSR,f           ;FSR (adresa počátku sektoru VRAM) + Offset -> FSR
        bcf     STATUS,C
        btfsc   INDF,6          ;nyní se do Carry flagu uloží hodnota konkrétního bitu z
příslušného sloupce
        bsf     STATUS,C
        rlf     IntTmp0,f       ;tento bit se "naroluje" do pomocného registru
        movf    Offset,f        ;kontrola, je-li offset již nula
        btfsc   STATUS,Z        ;jestli ano (Z==1) program pokračuje na změnu sektoru; jestli ne
(Z==0), program pokračuje v cyklu dále
        goto    NextSektor6
        decf    Offset,f        ;odečte od Offsetu jedničku
        goto    PokrRad6        ;a pokračuje na odečítání bitu
NextSektor6
        btfsc   Sektor          ;podle stavu bitu Sektor se zvolí následující sektor
        goto    NextSektor1_6   ;nyní je sektor dvě, změníme na jedničku
                                ;nebo je-li sektor v nule
        bsf     Sektor
        movlw   konecVRAM2
        movwf   Offset
PokrRad6
        decfsz  Bit,f           ;už byla odeslána osmice bitů?
        goto    Radek6          ;ještě ne, počítáme další bity
        goto    KonRad6
NextSektor1_6
        bcf     Sektor
```

```
        bsf     NaZacatek
        movlw   konecVRAM1
        movwf   Offset
KonRad6
        btfss   NaZacatek
        goto    BuffRad6
        bsf     Odeslano
        bcf     NaZacatek
        banksel PIE1
        bcf     PIE1,SSPIE
        banksel 0
BuffRad6
        movf    IntTmp0,w
        movwf   SSPBUF
        goto    ENDINT
;-------------------------------------------------------------------------------------------
Radek7
        movlw   VideoRAM1
        btfsc   Sektor          ;v případě, že Sektor==1, nastaví se počáteční adresa druhého
sektoru. Jinak se ponechá adresa prvního sektoru
        movlw   VideoRAM2
        movwf   FSR
        movf    Offset,w
        addwf   FSR,f           ;FSR (adresa počátku sektoru VRAM) + Offset -> FSR
        bcf     STATUS,C
        btfsc   INDF,7          ;nyní se do Carry flagu uloží hodnota konkrétního bitu z
příslušného sloupce
        bsf     STATUS,C
        rlf     IntTmp0,f       ;tento bit se "naroluje" do pomocného registru
        movf    Offset,f        ;kontrola, je-li offset již nula
        btfsc   STATUS,Z        ;jestli ano (Z==1) program pokračuje na změnu sektoru; jestli ne
(Z==0), program pokračuje v cyklu dále
        goto    NextSektor7
        decf    Offset,f        ;odečte od Offsetu jedničku
        goto    PokrRad7        ;a pokračuje na odečítání bitu
NextSektor7
        btfsc   Sektor          ;podle stavu bitu Sektor se zvolí následující sektor
        goto    NextSektor1_7   ;nyní je sektor dvě, změníme na jedničku
                                ;nebo je-li sektor v nule
        bsf     Sektor
        movlw   konecVRAM2
        movwf   Offset
PokrRad7
        decfsz  Bit,f           ;už byla odeslána osmice bitů?
        goto    Radek7          ;ještě ne, počítáme další bity
        goto    KonRad7
NextSektor1_7
        bcf     Sektor
        bsf     NaZacatek
        movlw   konecVRAM1
        movwf   Offset
KonRad7
        btfss   NaZacatek
        goto    BuffRad7
        bsf     Odeslano
        bcf     NaZacatek
        banksel PIE1
        bcf     PIE1,SSPIE
        banksel 0
BuffRad7
        movf    IntTmp0,w
        movwf   SSPBUF
        goto    ENDINT
;.....................................................................
; Přerušení od Timeru 0, obsluhuje spínání řádků
IntTMR
        bcf     INTCON,T0IF     ;vynuluje příznak
        bsf     OE              ;vypne výstup registrů
        goto    $+1             ;pauza 1us
        goto    $+1
        goto    $+1
        goto    $+1
        bsf     R0              ;zhasne řádky
        bsf     R1
        bsf     R2
        bsf     R3
```

```
        bsf     R4
        bsf     R5
        bsf     R6
        bsf     R7

        btfss   Odeslano        ;je-li příznak odeslano v jedničce, můžeme pokračovat
        goto    ENDINT          ;jinak ještě nebyl odeslán celý řádek do posuvných registrů
                                ;v podstatě by zde měla nastat nějaká chyba, ale tato situace by
se při správném časování rychlosti přerušení a portu neměla stát

        movlw   RozcestiInt / .256
        movwf   PCLATH
        bcf     STATUS,C
        rlf     Radek,w         ;vynásobí Radek dvěma a vloží jej do w
        addwf   PCL,f           ;skočí na řádek odpovídající obsluhovanému řádku
RozcestiInt
        bcf     R0              ;a tím sepne výstup
        goto    TmrPokr
        bcf     R1
        goto    TmrPokr
        bcf     R2
        goto    TmrPokr
        bcf     R3
        goto    TmrPokr
        bcf     R4
        goto    TmrPokr
        bcf     R5
        goto    TmrPokr
        bcf     R6
        goto    TmrPokr
        bcf     R7
TmrPokr
        bsf     LE
        bcf     OE
        incf    Radek,f
        movlw   .8
        xorwf   Radek,w
        btfsc   STATUS,Z        ;je-li řádek nyní roven osmi, Z bude v jedničce
        clrf    Radek
        bcf     Odeslano
        banksel PIE1
        bsf     PIE1,SSPIF
        banksel 0
        bcf     LE
        goto    ENDINT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Start programu, inicializace zařízení, vynulování výstupů a registrů
Start

        banksel SSPSTAT         ;registry v Bance 1
        movlw   B'10000011'     ;nastavení Timeru 0
        movwf   OPTION_REG

        clrf    TRISA           ;nastavení digitálních portů
        clrf    TRISB
        CLRF    TRISD
        CLRF    TRISE
        movlw   B'10010000'
        movwf   TRISC

        movlw   B'01000000'     ;nastavení synchronního sériového portu
        movwf   SSPSTAT

        movlw   B'00001000'
        movwf   PIE1
        clrf    PIE2

        banksel ANSEL           ;registry v Bance 3
        clrf    ANSEL           ;;nastavení analogových portů
        CLRF    ANSELH

        banksel 0               ;registry v Bance 0
        movlw   B'00100001'     ;nastavení synchronního sériového portu
        movwf   SSPCON

        movlw   B'01100000'
```

```
        movwf   INTCON
        clrf    PIR1
        clrf    PIR2

        bsf     R0
        bsf     R1
        bsf     R2
        bsf     R3
        bsf     R4
        bsf     R4
        bsf     R5
        bsf     R6
        bsf     R7
        bsf     OE
        bcf     LE

        clrf    Radek
        clrf    StatProg

        movf    SSPBUF,w
        clrf    SSPBUF

;       bsf     INTCON,GIE

        movlw   VideoRAM1
        movwf   TMP0
        bankisel        VideoRAM1
MazCykl1        movf    TMP0,w
        movwf   FSR
        clrf    INDF
        incf    TMP0,f
        movf    TMP0,w
        xorlw   konecVRAM1+1
        btfss   STATUS,Z
        goto    MazCykl1


        movlw   VideoRAM2
        movwf   TMP0
        bankisel        VideoRAM2
MazCykl2        movf    TMP0,w
        movwf   FSR
        clrf    INDF
        incf    TMP0,f
        movf    TMP0,w
        xorlw   konecVRAM2+1
        btfss   STATUS,Z
        goto    MazCykl2

        bsf     PIR1,SSPIF
        bsf     INTCON,GIE
        goto    $

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Hlavní smyčka - zde probíhá ukládání bitmap do VideoRAM
HlSmycka
        banksel EEADR           ;vloží adresu příslušné tabulky do registrů adresy
        movlw   SPS % 0ffh
        movwf   EEADR

        movlw   SPS / 0ffh
        movwf   EEADRH

        bankisel konecVRAM2
        movlw   konecVRAM2      ;vloží adresu videoram do fsr
        movwf   FSR

CyklusCteni
        banksel EECON1          ;čte z paměti
        bsf     EECON1,EEPGD
        bsf     EECON1,RD
        nop
        nop
        banksel EEDATH

        btfsc   EEDATH,0        ;kontroluje konec
```

```
        goto    ZmenaTextu
        movf    EEDATA,w                ;přepíše data do videoram
        movwf   INDF
        incf    EEADR,f         ;inkrementuje registr adresy paměti programu
        btfsc   STATUS,C
        incf    EEADRH,f        ;případně jeho vyšší byte
        decf    FSR,f           ;pak ještě adresu VideoRAM
        goto    CyklusCteni

ZmenaTextu
        bsf     INTCON,GIE
        goto    $

        goto    HlSmycka
;.....................................................
SPS
        data    B'00000000'
        data    b'00000000'
        data    b'01100101'
        data    b'10010110'
        data    b'10010101'
        data    b'10001000'
        data    b'00000000'
        data    b'00011000'
        data    b'00100100'
        data    b'00100100'
        data    b'11111100'
        data    b'00000000'
        data    b'01100100'
        data    b'10010100'
        data    b'10010100'
        data    b'10001000'
        data    3fffh

EI      data    b'00000000'
        data    b'00000000'
        data    b'00000000'
        data    b'00000000'
        data    b'00000000'
        data    b'00000000'
        data    b'10000000'
        data    b'11111010'
        data    b'10001000'
        data    b'00000000'
        data    b'10010000'
        data    b'10101000'
        data    b'10101000'
        data    b'10101000'
        data    b'01110000'
        data    b'00000000'
        data    3fffh
        end
```