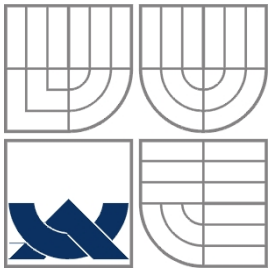
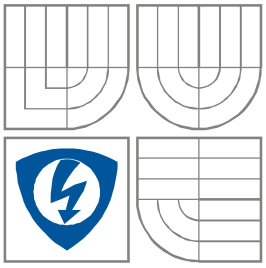


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

VGA ZOBRAZOVACÍ ZAŘÍZENÍ S MIKROKONTROLEREM

VGA IMAGE GENERATION WITH MICROCONTROLLER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jan Polášek
AUTHOR

VEDOUCÍ PRÁCE — Ing. Zbyněk Fedra, Ph.D.
SUPERVISOR

BRNO, 2008

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:	Jan Polášek
Bytem:	Budovatelská 6, Břeclav, 691 41
Narozen/a (datum a místo):	27. srpna 1983 v Kyjově

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 53, Brno, 602 00
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Dr. Ing. Zbyněk Raida, předseda rady oboru Elektronika a sdělovací technika
(dále jen „nabyvatel“)

Čl. 1

Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):

   ☐ disertační práce
   ☐ diplomová práce
   ☒ bakalářská práce
   ☐ jiná práce, jejíž druh je specifikován jako .....
   (dále jen VŠKP nebo dílo)

Název VŠKP:	VGA zobrazovací zařízení s mikrokontrolerem
Vedoucí/ školitel VŠKP:	Ing. Zbyněk Fedra, Ph.D.
Ústav:	Ústav radioelektroniky
Datum obhajoby VŠKP:	_____

   VŠKP odevzdal autor nabyvateli*:

   ☒ v tištěné formě – počet exemplářů: 2
   ☒ v elektronické formě – počet exemplářů: 2

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.

3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

* hodící se zaškrtněte

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.

2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.

3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti

   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy (z důvodu utajení v něm obsažených informací)

4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.

2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.

3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.

4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: 6. června 2008

……..	……
Nabyvatel	Autor

Abstrakt

Tato bakalářská práce se zabývá generováním obrazu formátu VGA pomocí mikrokontroleru. Je zaměřena na mikrokontrolery řady AVR od výrobce Atmel.

Práce obsahuje popis rozhraní VGA, řeší jeho připojení na výstupní porty mikrokontroleru, je zde uvedeno kompletní schéma zapojení výrobku zobrazovacího zařízení s mikrokontrolerem Atmel ATmega324P, který jsem zkonstruoval při řešení práce..

Je zde rozebrán princip generování obrazu v mikrokontroleru, synchronizace obrazu pomocí interního čítače mikrokontroleru, vytváření jednoduchých grafických obrazů a textu a princip generování barevného obrazu. Podrobně je rozebráno vykreslování textu na monitoru a vytváření matice znaků pro vykreslování včetně zdrojového kódu pro mikrokontroler.

Výsledkem této bakalářské práce je software pro mikrokontroler a funkční výrobek zobrazovacího zařízení.

Klíčová slova:
mikrokontroler, VGA, Atmel AVR, monitor, rozlišení, synchronizace

Abstract

This bachelor’s thesis is concerned with generating image in VGA format using microcontroller. It’s aimed at microcontrollers series AVR from producer Atmel.

Project contains VGA interface description, solves its connection to microcontroller’s output ports, further it contains connection diagram of image generation device with microcontroller Atmel ATmega324P, that I constructed during work on this project..

In project are analyzed fundamentals of image generating in microcontroller, synchronization of image using microcontroller’s internal counter, creating of simple graphic images and text and principles of generating colour images. Generation of text is analyzed in details, including source code for microcontroller.

The result of this bachelor‘s thesis is functional image generator device.

Keywords:
microcontroller, VGA, Atmel AVR, monitor, resolution, synchronization

POLÁŠEK, J. *VGA zobrazovací zařízení s mikrokontrolerem: bakalářská práce.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 34 s. Vedoucí bakalářské práce: Ing. Zbyněk Fedra, Ph.D.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma VGA zobrazovací zařízení s mikrokontrolerem jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

V Brně dne 6. června 2008 .....
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Zbyňku Fedrovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne 6. června 2008 .....
podpis autora

Obsah

1 Úvod

Cílem této bakalářské práce je prostudovat a objasnit možnost generování obrazu formátu VGA pomocí mikrokontroleru, navrhnout schéma zapojení obvodu pro generování obrazového signálu, vytvořit softwarové vybavení pro mikrokontroler a zkonstruovat výrobek zobrazovacího zařízení. Práce je zaměřena na mikrokontrolery řady AVR od výrobce Atmel, finální výrobek je osazen mikrokontrolerem Atmel ATmega324P.

První část práce je zaměřena na popis VGA rozhraní a objasnění jeho funkce, je zde vysvětlen princip vykreslování obrazu v monitoru. Také je zde popsán konektor pro připojení monitoru a vysvětleny funkce jednotlivých pinů. Další části se zabývají připojením monitoru na výstupní porty mikrokontroleru a tvorbu obrazu v mikrokontroleru. Je popsána synchronizace obrazu pomocí interního čítače a objasněn způsob, jakým je možné vytvářet grafiku a text pro zobrazení na monitoru. Generování textu je podrobně vysvětleno a je zde rozebráno získávání zobrazovaných dat přes jednotku sériového rozhraní USART. Dále práce obsahuje schéma zapojení přípravku zobrazovacího zařízení, který jsem zkonstruoval při řešení projektu. Závěrem práce je popsáno samotné programování a nastavení mikrokontroleru a software, který jsem použil při práci na projektu.

Součástí této práce jsou ukázky zdrojového kódu, které vysvětlují jednotlivé kroky tvorby obrazu v mikrokontroleru a nakonec je uveden kompletní zdrojový kód včetně komentářů.

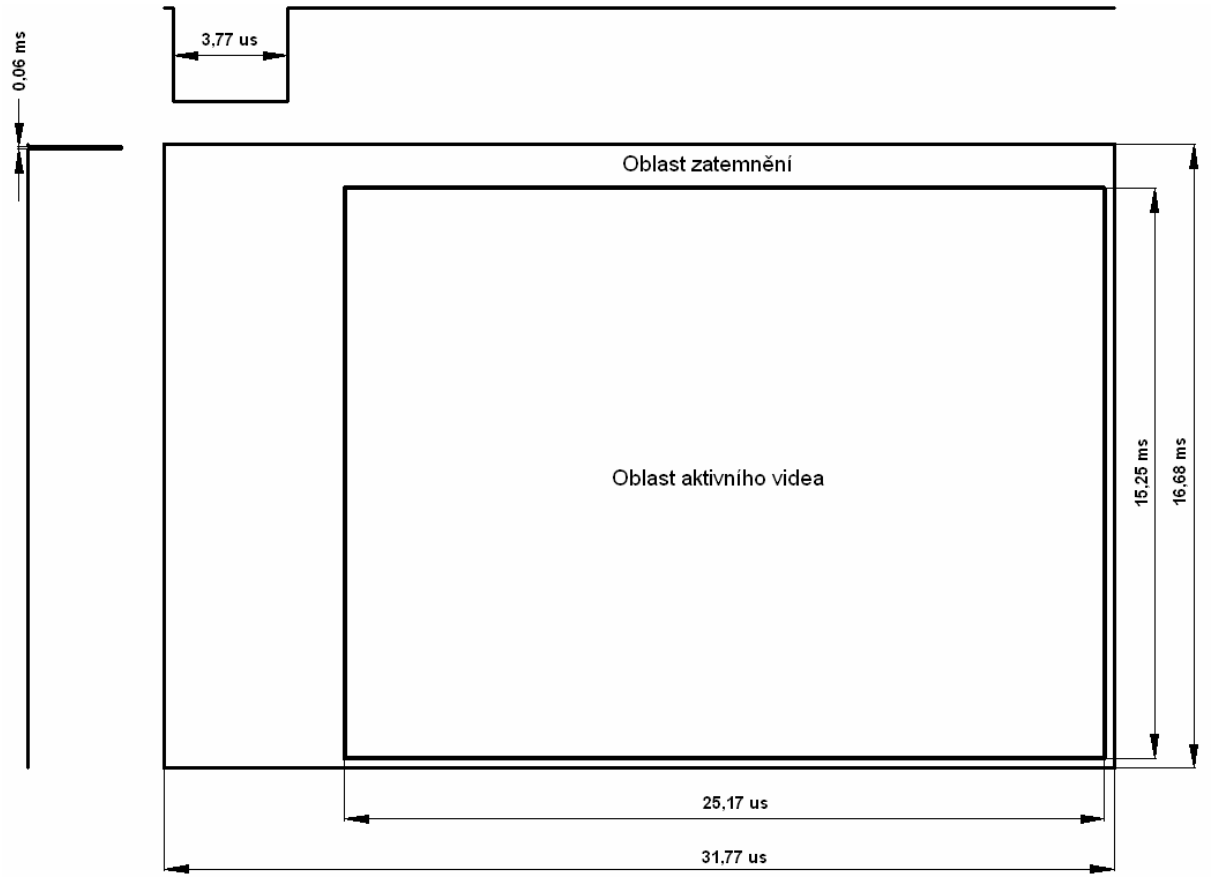
Výsledkem práce je funkční výrobek zobrazovacího zařízení s mikrokontrolerem Atmel ATmega324P včetně softwaru, který demonstruje možnosti tohoto zařízení.

2 Popis rozhraní VGA

Abychom vůbec mohli začít uvažovat nad generováním obrazu VGA jakýmkoliv zařízením, musíme nejdříve pochopit, na jakém principu systém tvorby obrazu pracuje.

VGA je standard pro analogové zobrazovací zařízení pro počítače PC, které bylo uvedeno firmou IBM v roce 1987. Formát obrazu je 640 x 480 / 60 Hz, což znamená, že obraz je složen ze 480 viditelných řádku a na každém řádku je zobrazeno 640 bodů. Obrazovka je překreslována 60x za sekundu, tedy přibližně jednou za 17 milisekund.

Obraz je na monitoru vykreslován po jednotlivých řádcích, na začátku každého řádku je poslán impuls horizontální synchronizace, dojde k vykreslení řádku a přejde se na další řádek. Jakmile je vykreslena celá obrazovka, je generován impuls vertikální synchronizace a začne se kreslit nový snímek. Mezi jednotlivými řádky i snímky jsou takzvané zatemňovací intervaly, v nich jsou právě vysílány synchronizační impulsy a paprsek vykreslující obraz na stínítku monitoru má čas přejít na začátek nového řádku, případně snímku. Pro lepší pochopení poslouží Obr. 1.



Obr. 1: Časování VGA signálu

Rozlišení obrazu je určeno z frekvencí horizontální a vertikální synchronizace a z polarity jejich signálů. Frekvence vertikální synchronizace udává zároveň obnovovací frekvenci obrazu, podíl frekvencí horizontální a vertikální synchronizace udává celkový počet

řádků. Tento počet je vetší než počet viditelných řádku, rozdíl je způsoben právě dobou potřebnou na synchronizační impuls a zpětný běh vykreslovacího paprsku. Horizontální rozlišení je odvozeno od počtu řádků a polarity synchronizačních impulsů. Pro rozlišení 640 x 480 jsou synchronizační impulzy aktivní v úrovni L. Pro signalizaci je použita TTL logika.

Co se bude vykreslovat na monitoru určují signály R, G, B, pro červenou, zelenou a modrou barvu. Charakteristická impedance vstupů těchto signálů je 75 Ω. Tyto signály mají úroveň 0 – 0,7 V a jejich výslednou kombinací se určuje výsledná barva každého bodu. Úroveň signálů se mění při průchodu řádkem pro každý bod zvlášť s bodovou frekvencí, která je pro režim VGA 25,175 MHz. Tyto signály jsou aktivní pouze v oblasti, která je na Obr. 1 označena jako „Oblast aktivního videa“. V oblasti zatemňovacího intervalu, na Obr. 1 označeném jako „Oblast zatemnění“, musí mít signály nulovou úroveň, protože v této oblasti se elektronika monitoru upíná k nulové úrovni napětí. Kdyby v této oblasti neměly barevné signálu nulovou úroveň, byl by obraz interpretován na monitoru nesprávně nebo vůbec.

Zatemňovací interval je rozdělen na několik oblastí: front porch, synchronizační impuls a back porch, dále je z každé strany okolo viditelného obrazu 8 pixelů okraje, které také patří do zatemňovacího intervalu. Celkový počet řádků (aktivní video + zatemnění) je pro režim VGA 525, v každém řádku je 800 bodů. Vše je shrnuto v Tab. 1.

Bodová frekvence	25,175	MHz
Řádková frekvence	31,469	kHz
Obrazová frekvence	59,94	Hz
Doba trvání bodu	39,7	ns
Doba řádku	31,77	µs
Doba snímku	16,7	ms

1 řádek	8	bodů front porch	0,3 µs
	96	bodů horizontální synchronizace	3,77 µs
	40	bodů back porch	1,6 µs
	8	bodů levý okraj	0,3 µs
	640	bodů aktivní video	25,17 µs
	8	bodů pravý okraj	0,3 µs
celkem 800 bodů na řádek			
1 obraz	2	řádky front porch	63,5 µs
	2	řádky vertikální synchronizace	63,5 µs
	25	řádků back porch	794 µs
	8	řádků horní okraj	254 µs
	480	řádků aktivní video	15,25 ms
	8	řádků dolní okraj	254 µs
celkem 525 řádků na obraz			

Tab. 1: Shrnutí charakteristiky VGA signálu

Postupným vývojem bylo dosaženo obrazových standardů s vyšším rozlišením a větší obnovovací frekvencí, jako například SVGA 800 x 600 bodů, XGA 1024 x 768 bodů a další s ještě lepšími parametry, těmito se však nebudeme dále zabývat, protože jejich generování pomocí mikrokontrolerů řady AVR není v uspokojivé kvalitě možné vzhledem k rostoucím frekvencím.

Monitor se připojuje patnáctipinovým konektorem označovaným jako D-SUB. Zapojení konektoru je v Tab. 2, převzato z [1].

Pin	Název	Popis
1	RED	červený video kanál
2	GREEN	zelený video kanál
3	BLUE	modrý video kanál
4	ID2	monitor ID bit 2 nebo nezapojen
5	GND	zem
6	RGND	zem pro červený video kanál
7	GGND	zem pro zelený video kanál
8	BGND	zem pro modrý video kanál
9	+5V	napájení 5V nebo nezapojen
10	SGND	zem pro synchronizační signály
11	ID0	monitor ID bit 0
12	ID1 nebo SDA	SDA nebo monitor ID bit 1
13	HSYNC	horizontální synchronizace
14	VSYNC	vertikální synchronizace
15	ID3 nebo SCL	SCL nebo monitor ID bit 3

Tab. 2: Zapojení D-SUB konektoru

Pro účely tohoto projektu jsou v zapojení důležité pouze barvonosné kanály R, G, B, dále horizontální a vertikální synchronizace a zem, krátce se však zmíním i o funkci dalších pinů.

Dříve se na pinech 4, 11, 12 a 15 používaly tzv. ID bity, které byly uvnitř monitoru buď spojené se zemí nebo nezapojené a jejich kombinace určovala, zda jde o černobílý nebo barevný monitor a jaké zvládá rozlišení. Na dnešních moderních monitorech již tyto bity nejsou, ale je místo nich zapojena na pinech 12 a 15 standardní sběrnice I_2C, po které spolu můžou počítač a monitor komunikovat. Jde zejména o zjištění maximálního rozlišení a obnovovací frekvence, které monitor zvládá a dále je možné nastavovat parametry monitoru přímo z PC pomocí ovladače od výrobce monitoru, pokud to daný monitor umožňuje.

3 Propojení monitoru a mikrokontroleru

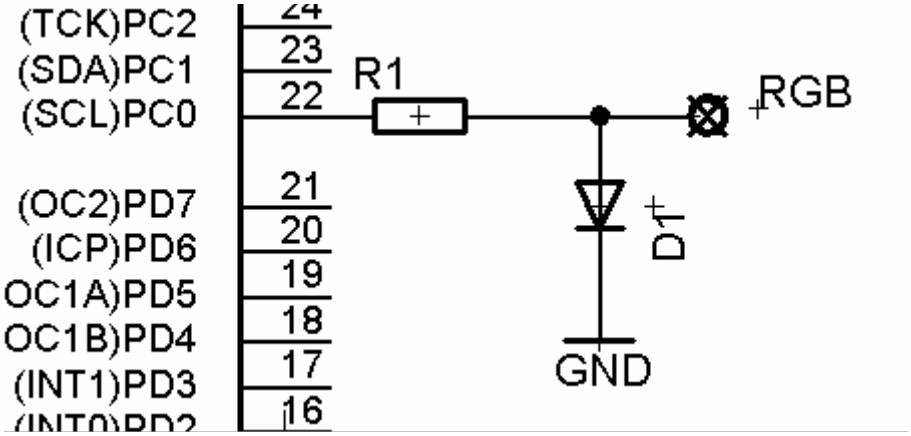
Signály pro horizontální a vertikální synchronizační impulsy mají úrovně TTL, je tedy možné připojit jejich vstupy na konektoru přímo na výstupní porty mikrokontroleru.

Signály barevných vstupů R, G, B mají maximální úroveň 0,7 V, není tedy možné připojit je přímo na výstupní port, ale signál je třeba přizpůsobit. Nejjednodušším způsobem je použití předřadného rezistoru. Uvážíme-li, že charakteristická impedance barevného vstupu R_{IN} = 75 Ω, maximální napětí na vstupu U_{VST} = 0,7 V a napětí na výstupním portu mikrokontroleru v úrovni H je U_H = 5 V, tak požadovaná hodnota předřadného rezistoru je

$$R = \frac{U_H - U_{VST}}{\frac{U_{VST}}{R_{IN}}} = \frac{5 - 0{,}7}{\frac{0{,}7}{75}} = 460{,}7\ \Omega\,. \qquad (1)$$

Tato hodnota není kritická, při použití rezistoru o této velikosti je obraz poměrně tmavý. Je vhodné použít hodnotu nižší, aby měl obraz větší jas, mám vyzkoušenu dobrou funkci při hodnotě rezistoru R = 250 Ω.

Jiný způsob jak přizpůsobit signál pro barevný vstup, je zapojit z výstupního portu přes rezistor proti zemi diodu pólovanou v propustném směru. Signál pro monitor se bude odebírat z této diody. Toto zapojení je uvedeno v [2]. Bude-li výstupní port v úrovni L, bude na signálu pro monitor napětí 0 V, bude-li port v úrovni H, na diodě se vytvoří úbytek napětí o velikosti asi 0,7 V. Je vhodné vybrat diodu, která má velký úbytek napětí, aby nebyl obraz na monitoru příliš tmavý. Zapojení je naznačeno na Obr. 2. Tento způsob zapojení jsem testoval, ale vrátil jsem se k zapojení s předřadným rezistorem, protože většina běžných diod má úbytek napětí v rozmezí 0,5 až 0,6 V a tak obraz na monitoru neměl plný jas.



Obr. 2: Napojení barevného signálu na výstupní port

4 Generování obrazu v mikrokontroleru

4.1 Generování synchronizačních impulsů

Pro generování VGA obrazu je klíčové dodržet frekvenci synchronizačních impulsů. Kritická je zejména pravidelnost impulsů. Proto se pro tvorbu impulsů použije přerušení od interního čítače.

Synchronizační rutina je tvořena tak, že interní čítač generuje přerušení s řádkovou frekvencí 31,469 kHz. Je zavedena proměnná, která počítá řádky, to je potřeba pro generování horizontálního synchronizačního impulsu, který je aktivní po dobu trvání třetího a čtvrtého řádku. Na začátku každého přerušení se přičte 1 k proměnné počítající řádky a je generován horizontální synchronizační impuls, který trvá asi 3,77 µs. Pokud je počet řádků roven 525, tak je proměnná vynulována.

Ukázka zdrojového kódu synchronizační rutiny:

```
// generování synchronizačních impulsů
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 20000000UL    // frekvence CPU pro delay.h
#include <util/delay.h>
volatile unsigned int radek; //aktuální řádek
ISR (TIMER0_OVF_vect)
{
    TCNT0 = 179;
    if (++radek == 525)
          radek = 0;
//VS 2 VGA řádky, zároveň se nastaví počátek HS,
synchronizace je aktivní v úrovni L
    if ((radek == 2 )||(radek == 3 ))
    {
          PORTD=0x00;
    }
    else
    {
          PORTD=0x40;
          _delay_us ( 3 );
          PORTD |= (1 << PORTD7); //vypnutí HS
    }
}
int main(void)
{
sei();
DDRD = 0xC0;
PORTD |= (1 << PORTD7);
PORTD |= (1 << PORTD6);
TIMSK0 = 0x01;
TCCR0B = 0x02;
TCNT0 = 179;
for(;;){}
}
```

Tento program obstará pravidelné generování synchronizačních impulsů. Vstup monitoru pro horizonální synchronizaci se připojí na PORTD7 a pro vertikální na PORTD6.

Dojde k zapnutí monitoru, na obrazovce nebude nic vidět, ale v menu bude patrné, že do monitoru putují správné synchronizační impulsy pro rozlišení VGA 640 x 480 / 60 Hz, jak je vidět na Obr. 3.

Obr. 3: Menu monitoru po přivedení synchronizačních impulsů

V programu je nastaveno povolení přerušení od interního čítače 0, nastavena předdělička 1/8 (podle návodu v [3]) a do registru TCNT0, který obsahuje aktuální hodnotu čítače, je nastavena hodnota 179. Tato hodnota musí být nastavena do registru znovu při každém vykonání přerušení. Takové nastavení čítače zajistí generování správné řádkové frekvence.

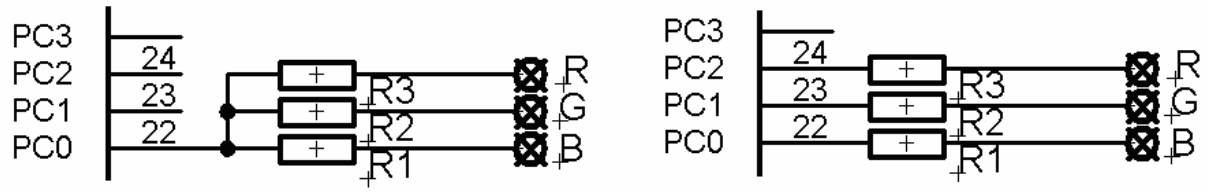
Mikrokontroler pracuje na frekvenci 20 MHz. Při nastavení čítače s děličkou 1/8 pracuje čítač s frekvencí 2,5 MHz. K jeho přetečení dochází s frekvencí 9765,625 Hz. Pro generování vertikálního synchronizačního pulsu potřebujeme frekvenci 31469 Hz, proto je nutné do registru TCNT0 vkládat počáteční hodnotu čítače.

Když není v registru TCNT0 žádná počáteční hodnota, tak čítač počítá do 255, my potřebujeme, aby počítal do 80, protože 2,5 MHz / 31469 Hz = 79,4. Tomu by odpovídala počáteční hodnota TCNT0 = 255 – 80 = 175, ale při této hodnotě je obnovovací frekvence monitoru nízká, asi 57 Hz, experimentálně jsem zjistil, že nejvhodnější nastavení je 179, takto se obnovovací frekvence monitoru nejvíce blíží 60 Hz.

4.2 Generování jednoduché grafiky

Když umíme generovat synchronizační impulsy, můžeme se dále zabývat jednoduchou grafikou. Vstupy R, G, B jsou ovládány z výstupních portů přes přizpůsobovací obvod, jak bylo uvedeno v kapitole 3.

Je možné vytvářet monochromatický obraz, v takovém případě spojíme všechny tři barevné vstupy a připojíme je na jeden výstupní port. Je možné také vytvářet barevný obraz, v tom případě zapojíme každý z barevných vstupů na samostatný port. Zapojení je naznačeno na obr. 4.



Obr. 4: Zapojení pro monochromatický a barevný obraz

Při použití zapojení pro barevný obraz můžeme získat signál, který má 8 barev. Barevnost obrazu bude takto omezena, protože nebudeme využívat rozmezí 0 až 0,7 voltu pro barevnou složku, ale pouze krajní hodnoty, tedy buď daná barva bude vypnutá nebo bude svítit naplno. Bude možné použít barvy, které vzniknou kombinací základních barev R, G, B podle Tab. 3.

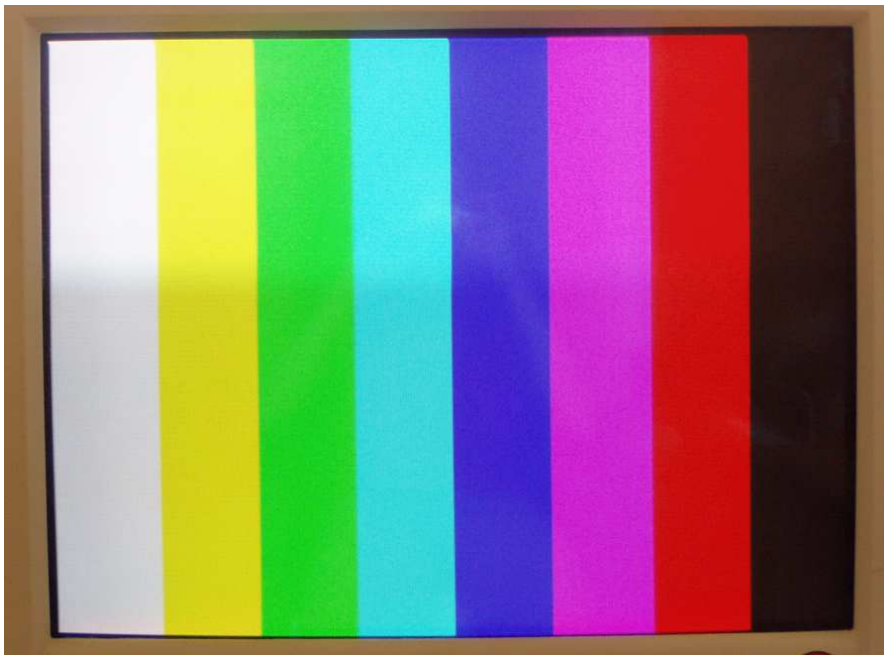
R	G	B	barva
1	1	1	bílá
1	1	0	žlutá
1	0	1	fialová
1	0	0	červená
0	1	1	tyrkysová
0	1	0	zelená
0	0	1	modrá
0	0	0	černá

Tab. 3: vliv kombinace RGB na barvu bodu

V oblasti, která je označena na Obr. 1 jako „Oblast aktivního videa“ můžeme vytvářet na monitoru obraz přiváděním vhodných signálů na vstupy R, G, B.

Můžeme vykreslit např. čáru přes celý řádek, to provedeme tak, že na dobu aktivního videa nastavíme výstupní port do úrovně H. Pokud chceme kreslit vertikální čáru, je postup poněkud složitější, musíme v každém řádku obrazu na dobu trvání několika bodů nastavit výstupní port do úrovně H a hned zpět do úrovně L. Pokud chceme nakreslit vertikální čáru dlouhou pouze několik řádků, zapínáme port do úrovně H pouze v těchto řádcích. Tímto postupem můžeme vytvářet jednoduchou čárovou grafiku. Můžeme libovolně kombinovat i barvy. Složitost obrazu je omezena výpočetním výkonem procesoru. Pokud uvážíme, že bodová frekvence je 25,175 MHz a frekvence procesoru je např. 20 MHz, tak je jasné, že není možné vykreslit na monitoru samostatný jeden bod, vykreslit se dá pouze krátká čárka dlouhá asi 2 až 5 bodů podle frekvence použitého mikrokontroléru.

Při pokusech s vytvářením jednoduché grafiky jsem zkoušel vykreslovat na monitoru různé konfigurace barevných pruhů, na Obr. 5 je vyfocen monitor, na kterém je zobrazena „duha“ z barevných pruhů osmi barev, které jsou uvedeny v Tab. 3.



Obr. 5: Duha z barevných pruhů

Horizontální čára ve střední části monitoru, která je vidět na Obr. 5 na monitoru zobrazena není, vznikla při focení a je způsobena odlišnou obnovovací frekvencí monitoru a snímacího čipu ve fotoaparátu.

Ukázka zdrojového kódu vytváření vertikálních duhových pruhů:

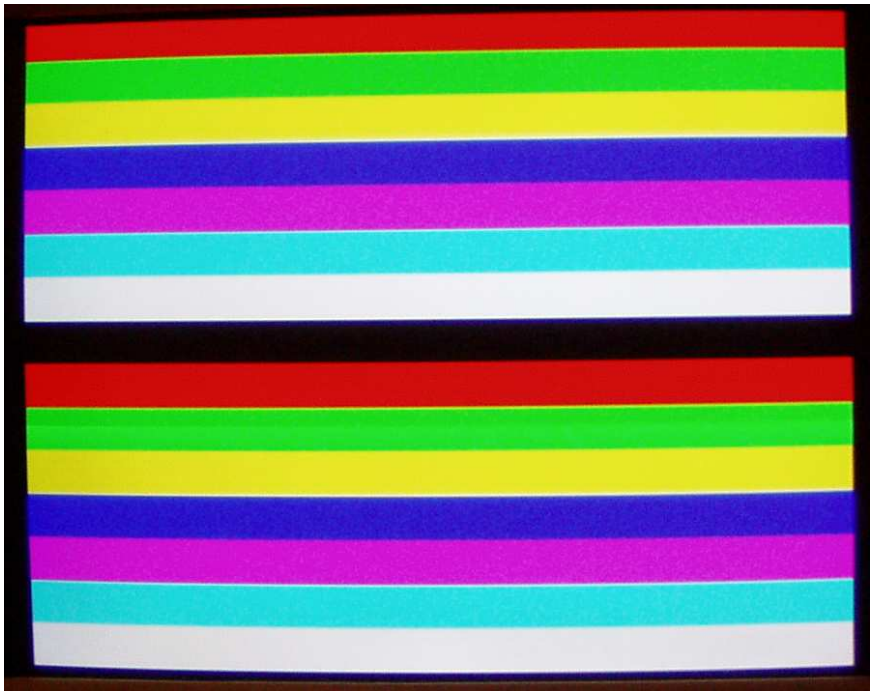
```
#define pic_off PORTC=0
#define cekej 2.6

if ((radek < 37)||(radek > 517))
{
    pic_off;
}
else
{
    _delay_us ( cekej );
    PORTC=7;
    _delay_us ( cekej );
    PORTC=3;
    _delay_us ( cekej );
    PORTC=2;
    _delay_us ( cekej );
    PORTC=6;
    _delay_us ( cekej );
    PORTC=4;
    _delay_us ( cekej );
    PORTC=5;
    _delay_us ( cekej );
```

```
        PORTC=1;
        _delay_us ( cekej );
        pic_off;
        _delay_us ( cekej );
}
```

Barevné vstupy monitoru R, G, B, jsou zapojeny na výstupy portu C 0, 1 a 2 mikrokontroleru. Po uplynutí impulsu horizontální synchronizace dojde k vyhodnocení, jestli současný řádek patří do oblasti aktivního videa nebo do oblasti zatemnění. Když patří řádek do oblasti aktivního videa, tak se začnou na řádku vykreslovat barevné čárky, každá s dobou trvání přibližně 2,6 µs. Barva čárky je určena hodnotou na portu C, tedy tím, které barevné vstupy monitoru jsou zrovna aktivní. Na konci řádku je obsah portu vynulován, aby nebyl žádný barevný vstup aktivní v oblasti zatemnění. Toto se opakuje pro všechny viditelné řádky obrazu.

Jiným způsobem je možné vytvářet horizontální barevné pruhy, jak je vidět na Obr. 6. Opět je provedeno vyhodnocení, zda se aktuální řádek nachází v oblasti aktivního videa, a pokud ano, tak se do výstupního registru portu zapisuje číslo aktuálního řádku dělené 32. Číslo aktuálního řádku se samozřejmě může dělit jakýmkoliv číslem, od toho se odvíjí šířka pruhů, ale je vhodné ho dělit číslem, které je mocninou 2. Pokud je v mikrokontroleru prováděno dělení číslem, které je mocninou 2, tak se to provádí bitovým posuvem, namísto klasického dělení, protože je to mnohem rychlejší. Pokud dělitel není mocninou 2, tak operace dělení trvá podstatně déle a na začátku vykreslovaných řádků vzniká tmavé místo.



Obr. 6: Horizontální barevné pruhy

Ukázka zdrojového kódu vytváření horizontálních duhových pruhů:

```
#define pic_off PORTC=0

if ((radek < 37)||(radek > 517))
{
    pic_off;
}
else
{
    PORTC=radek/32;
    pic_off;
}
```

4.3 Generování textu

Pro zobrazování textu na monitoru je třeba vytvořit a do paměti mikrokontroleru umístit znakovou sadu v podobě matic s jednotlivými znaky.

Jednotlivé znaky jsou v paměti umístěny tak, že každý znak je rozložen na jednotlivé řádky umístěné v paměti za sebou. Z tohoto důvodu je každý znak sady široký 8 bodů. Výška matice znaku je 12 bodů. Matice znaku má z jedné strany v sobě obsaženu mezeru, která slouží jako mezera mezi jednotlivými znaky. Stejně tak má matice v sobě obsaženy mezery z vrchu a ze spodu, které slouží jako mezera mezi jednotlivými řádky textu. Toto opatření dovolí skládat znaky při vykreslování hned za sebe, aniž by mezi ně musely být vkládány mezery, a přesto jsou vykreslené znaky na obrazovce od sebe odděleny. Na jeden znak v matici zbývá tedy maximálně 7 bodů na šířku a 10 bodů na výšku. Signál je na výstupní port zapisován jako řetězec tvořený řádky těchto matic jdoucích za sebou.

Znaková sada obsahuje velká a malá písmena, číslice a speciální znaky, které jsou obsaženy na anglické klávesnici. Znaky jsou v paměti uloženy za sebou podle pořadí, jaké mají v tabulce ASCII.

```
11111111
00000011
01111101
01111101
01111101
00000011
01111101
01111101
01111101
00000011
11111111
11111111
```

Obr. 7: symbol B znakové sady

Na Obr. 7 je zakreslen příklad symbolu B, jak je vytvořen ve znakové sadě, písmeno má šířku 7 bodů, vpravo je mezera.

Celý princip generování textu funguje tak, že je vytvořeno pole, které má takovou velikost, jako je maximální počet znaků, který je možné zobrazit na obrazovce. V tomto poli jsou uloženy kódy jednotlivých znaků v takovém pořadí, v jakém jsou znaky zobrazeny na obrazovce. Kódy znaků jsou ve formátu ASCII. Funkce, která se stará o vykreslování textu, si z tohoto pole bere adresy znaků ze znakové sady.

Generování obrazu probíhá tak, že do výstupního registru portu je načten řádek matice znaku, jehož adresu získá vykreslovací funkce z pole znaků. Poté se provede sedmkrát po sobě operace bitového posuvu vlevo, znak je takto „vysunut“ z portu. Poté je načten do výstupního registru portu řádek matice dalšího znaku, který následuje na řádku textu, opět se provede bitový posuv a tak se pokračuje až do vykreslení jednoho řádku obrazu.

Jsou zavedeny proměnné „akt_radek_textu“ a „akt_radek_znaku“. Proměnná „akt_radek_textu“ udává, kolikátý řádek textu je právě vykreslován a proměnná „akt_radek_znaku“ udává, kolikátý je vykreslován řádek matice znaku. Tato proměnná funguje jako ukazatel na aktuální vykreslovaný řádek matice znaku. Je inkrementována s každým řádkem obrazu, dokud nejsou vykresleny všechny řádky matice znaku. Poté je proměnná vynulována a začne se kreslit další řádek textu.

Kvůli způsobu „vysouvání“ znaku z portu musí být signál monochromatický, všechny tři barevné vstupy monitoru jsou zapojeny na jeden pin výstupního portu, jak je naznačeno na Obr. 4 v zapojení pro monochromatický obraz. Signál je odebírán z posledního pinu výstupního portu.

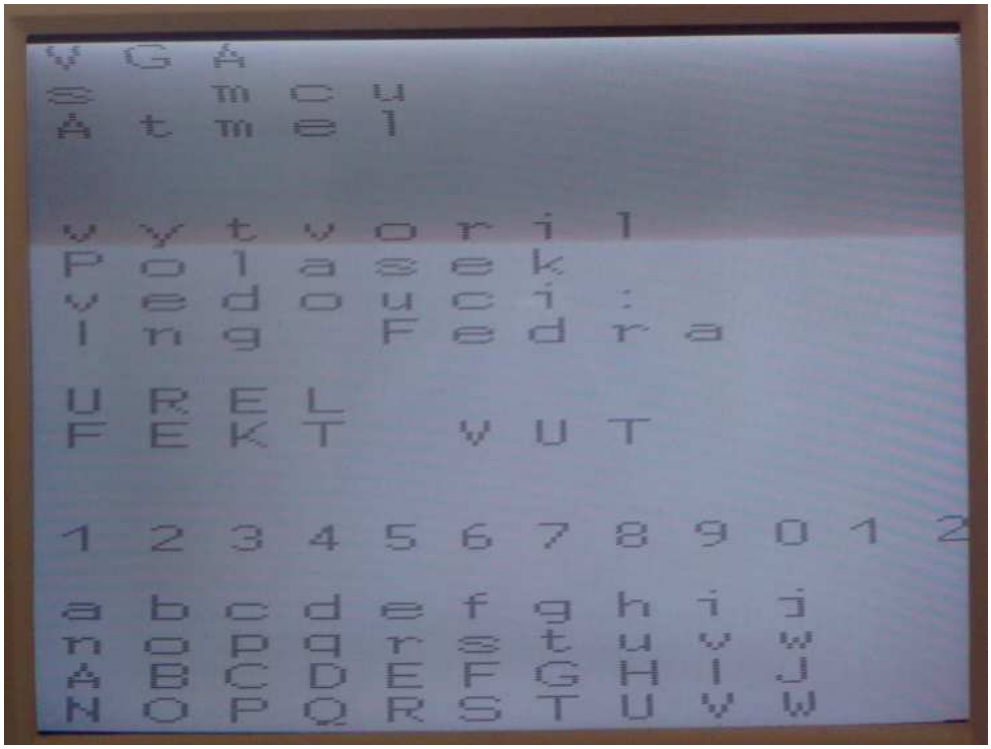
Toto řešení má velkou nevýhodu v tom, že pro operaci bitového posuvu a tedy vysunutí řádku matice z výstupního portu je spotřebováván strojový čas, a není možné v té době provádět nic dalšího. Tímto je značně omezen počet zobrazených znaků na řádek a mezi znaky vznikají velké mezery, protože po vysunutí řádku matice z portu je potřeba do portu načíst řádek matice znaku, který následuje na řádku za naposledy vykresleným znakem, a v této době neprobíhá vykreslování.

Významného vylepšení a zvýšení počtu znaků na řádek je možné dosáhnout použitím výstupu MOSI rozhranní SPI pro vykreslování textu. Tento poznatek jsem získal v [4]. Celý algoritmus je stejný jako při použití výstupního portu až po operaci načtení řádku matice do výstupního registru portu. Místo do výstupního registru portu je řádek matice znaku načten do datového registru SPDR rozhraní SPI. Z tohoto registru jsou data po načtení automaticky vysouvána na pinu MOSI, aniž by tím byl spotřebováván strojový čas. Neztrácí se tedy čas mnoha operacemi bitového posuvu uvnitř registru a do registru SPDR můžeme postupně načítat řádky matic znaků, které jdou po sobě na řádku textu, aniž bychom se museli starat o vysouvání.

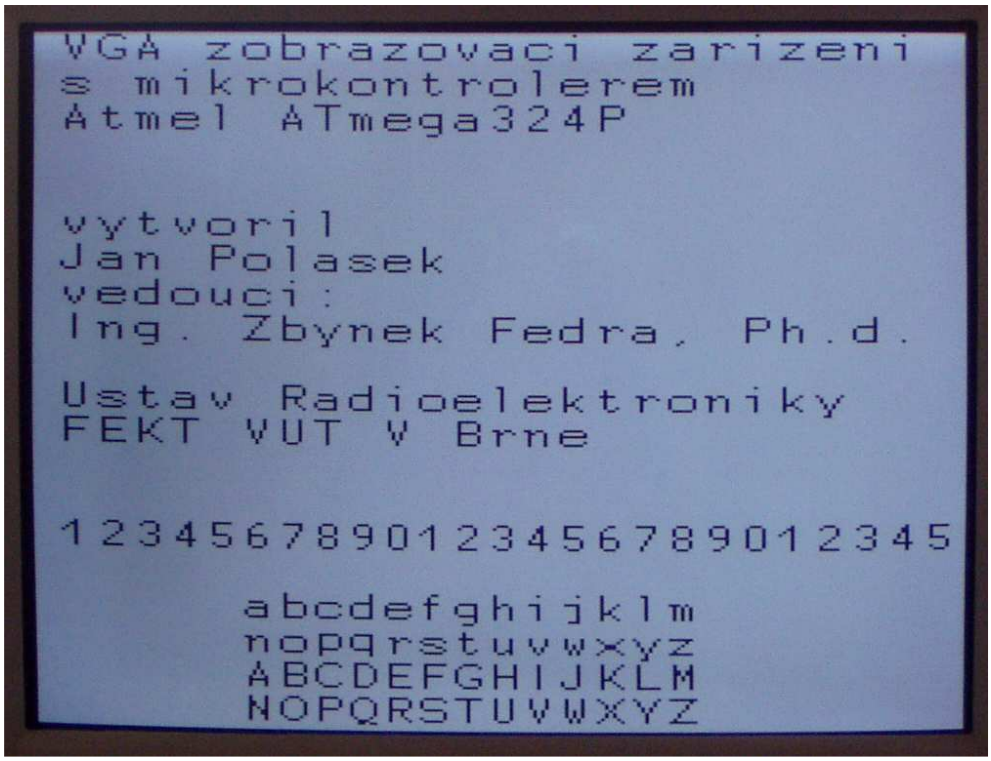
Maximální rychlost vysouvání dat z registru SPDR na pin MOSI je rovna poloviční taktovací frekvenci mikrokontroleru, při použití mikrokontroleru s frekvencí 20 MHz je tedy maximální rychlost změny na pinu MOSI 10 MHz.

Počet znaků na jeden řádek textu při vykreslování přes pin MOSI rozhraní SPI je 25, při použití vykreslování z portu je počet znaků na řádek 12. Výška jednoho znaku v matici je 12 bodů, ale každý řádek matice je kreslen dvakrát, písmo má tak na monitoru přirozený tvar. Kdybychom vykreslovali každý řádek jednou, písmo by bylo příliš roztaženo do šířky. Takto je výška jednoho znaku na obrazovce 24 obrazových řádků. Počet zobrazitelných řádků textu je 480 / 24, tedy 20. Pole znaků má tedy velikost 25 * 20, což je 500 znaků.

Rozdíl mezi použitím vykreslování přes výstupní registr portu a přes pin MOSI je dobře patrný při porovnání obrázků Obr. 8 a Obr. 9.



Obr. 8: Obraz získaný vykreslováním přes výstupní registr portu



Obr. 9: Obraz získaný vykreslováním přes pin MOSI

5 Získávání zobrazovaných dat

Abychom mohli pomocí mikrokontroleru zobrazovat data, musíme je do mikrokontroleru nejdříve nějakým způsobem dostat. Nejvhodnějším způsobem získávání zobrazovaných dat se jeví být zasílání dat do mikrokontroleru přes některou z komunikačních sběrnic, které mikrokontroler obsahuje. Mikrokontrolery Atmel řady AVR obsahují sběrnice SPI, I_2C a sériové rozhraní USART.

K získávání dat pro zobrazení jsem zvolil sériové rozhranní USART. Hlavním důvodem bylo, že narozdíl od sběrnic SPI a I_2C je sériové rozhraní běžnou součástí osobních počítačů, což umožnilo snadné testování při práci na projektu. Sběrnici SPI nebylo možné použít také z důvodu, že pin MOSI je používán k zobrazování. Drobnou nevýhodou tohoto řešení je to, že napěťové úrovně sériového rozhraní USART v mikrokontroleru se liší od úrovní používaných sériovým rozhraním RS232 v osobních počítačích, takže je nutné použít převodník. Použil jsem jednoduchý a levný převodník s integrovaným obvodem MAX232 od výrobce Maxim. Konstrukce tohoto převodníku je popsána v [5].

Mikrokontroler ATmega324P disponuje dvěma sériovými jednotkami USART, označenými USART0 a USART1. Pro přijímání zobrazovaných znaků jsem použil jednotku USART0. Pro správnou funkci je zapotřebí nastavit počet datových bitů, přenosovou rychlost a případně paritu. Toto nastavení se provádí pomocí registrů UCSR0B, UCSR0C a UBRR0, podrobnosti jsou uvedeny v datasheetu mikrokontroleru ATmega324P [6]. Použité nastavení je 8 datových bitů, žádná parita, přenosová rychlost 9600 b/s. Přenosová rychlost se nastavuje v registru UBRR0, pro hodnotu registru platí vztah (2), kde f_0 je taktovací rychlost mikrokontroleru a f_{BR} je přenosová rychlost.

$$UBRR = \frac{f_0}{16 \cdot f_{BR}} - 1 = \frac{20000000}{16 \cdot 9600} - 1 = 130. \qquad (2)$$

Testoval jsem přenosové rychlosti od 9600 b/s až po 28800 b/s. Mezi rychlostmi jsem z uživatelského hlediska nezaznamenal žádné rozdíly, zobrazení znaku při přenosu je při všech přenosových rychlostech okamžité, proto jsem zvolil nízkou přenosovou rychlost, což umožňuje přenos dat na velkou vzdálenost a velkou odolnost proti rušení.

Data jsem do mikrokontroleru posílal pomocí programu Terminal, o kterém se ještě zmíním v kapitole 8.

Pomocí sériového rozhraní jsou do mikrokontroleru posílány adresy znaků ve formátu ASCII, které jsou ukládány do pole znaků, odkud si je načítá vykreslovací funkce pro zobrazování.

Je zavedena proměnná „cislo_akt_znaku“, která vyjadřuje pořadí znaku, který byl naposledy přijat, v poli znaků. Po příjmu znaku je vyhodnoceno, jestli přijaty znak není Backspace nebo Enter. Když je přijat Backspace, dojde ke smazání posledního přijatého znaku v poli znaků, když je přijat Enter, provede se přechod na nový řádek. (hodnota proměnné „cislo_akt_znaku“ se navýší tak, aby se příští přijatý znak zobrazil na novém řádku). Pokud je přijat jiný znak než Backspace nebo Enter, zapíše se jeho adresa na aktuální pozici v poli znaků a proměnná „cislo_akt_znaku“ je inkrementována o 1. Dojde-li k tomu, že je pole znaků plné (obrazovka je už celá zaplněna znaky), je po příjmu nového znaku vynulováno, je vynulována také proměnná „cislo_akt_znaku“ a nově přijatý znak se vypíše na začátek obrazovky.

6 Zdrojový kód a rozbor programu

V této kapitole je uveden kompletní zdrojový kód programu zobrazovacího zařízení. Kód je bohatě komentován, přesto je za výpisem kódu uvedeno detailní vysvětlení funkce programu. Program je ve zdrojové i zkompilované formě obsažen na přiloženém CD.

6.1 Výpis zdrojového kódu programu

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <avr/pgmspace.h>
#include <string.h>
#include "matice_znaku.h"

#define NOP asm("nop")
//parametry obrazu
#define vga_pocet_radku 525
#define znaku_na_radek 25
#define radku_textu 20
#define celkem_znaku 500
#define vyska_znaku 24
#define vs_on   PORTD=0x00
#define vs_off  PORTD=0x40
#define hs_off PORTD |= (1 << PORTD7)
#define zapni_vykreslovani  DDRB=0xB0
#define vypni_vykreslovani  DDRB=0x90

volatile unsigned char pole_znaku[znaku_na_radek*radku_textu];
volatile unsigned int cislo_akt_znaku;
volatile unsigned char povol_vykreslovani;
volatile unsigned char akt_radek_textu;
volatile unsigned int radek;
volatile unsigned char akt_radek_znaku;

//uvodni obrazovka po zapnuti, retezce ulozeny v programove pameti
const char str1[] PROGMEM =   "VGA zobrazovaci zarizeni";
const char str2[] PROGMEM =   "s mikrokontrolerem";
const char str3[] PROGMEM =   "Atmel ATmega324P";
const char str4[] PROGMEM =   "vytvoril";
const char str5[] PROGMEM =   "Jan Polasek";
const char str6[] PROGMEM =   "vedouci:";
const char str7[] PROGMEM =   "Ing. Zbynek Fedra, Ph.d.";
const char str8[] PROGMEM =   "Ustav Radioelektroniky";
const char str9[] PROGMEM =   "FEKT VUT V Brne";
const char str10[] PROGMEM =   "123456789012345678901234";
const char str11[] PROGMEM =   "     abcdefghijklm";
const char str12[] PROGMEM =   "     nopqrstuvwxyz";
const char str13[] PROGMEM =   "     ABCDEFGHIJKLM";
const char str14[] PROGMEM =   "     NOPQRSTUVWXYZ";

static void ini(void);
```

```
//synchronizacni rutina
ISR (TIMER0_OVF_vect)
{
	TCNT0 = 0xB3;
	//pocitani radku, na konci snimku vynulovani promennych
	if (++radek == vga_pocet_radku)
	{
		radek = 0;
		akt_radek_textu = 0;
		akt_radek_znaku = 0;
	}
	//VS dva radky, zaroven se nastavi pocatek HS
	if ((radek == 2 )||(radek == 3 ))
	{
		vs_on;
	}
	else
	{
		vs_off;
	}
	//povoleni vykreslovani v oblasti aktivniho videa
	//nez se toto provede, trva impuls HS
	if ((radek < 37)||(radek > 517))
	{
		povol_vykreslovani = 0;
	}
	else
	{
		povol_vykreslovani = 1;
		//pocitani radku textu
		if (++akt_radek_znaku == vyska_znaku)
		{
			akt_radek_textu++;
			akt_radek_znaku = 0;
		}
		else
		{
			NOP;
			NOP;
			NOP;
			NOP;
			NOP;
			NOP;
			NOP;
			NOP;
		}
	}
	hs_off;
}
//inicializace a nastaveni
static void ini(void)
{
	//nastaveni SPI
	DDRB=0xB0;
	SPSR = 1 << SPI2X;
	SPCR = (1 << SPE) | (1 << MSTR);
	//nastaveni UART, 8 datovych bitu, rychlost 9600 b/s
	UCSR0B = 0x00;
	UCSR0A = 0x00;
	UCSR0C = 0x86;
	UBRR0L = 0x82;
```

```
	UBRR0H = 0x00;
	UCSR0B = (1 << RXEN0);
	//nastaveni idle rezimu
	set_sleep_mode(SLEEP_MODE_IDLE);
	//nastaveni portu D pro synchronizaci
	DDRD = 0xC0;
	PORTD |= (1 << PORTD7);
	PORTD |= (1 << PORTD6);
	//nastaveni casovace pro radkovou synchronizaci
	TCNT0 = 0xB3;
	TCCR0B = 0x02;
	TIMSK0 = 0x01;
	//povoleni globalniho preruseni
	sei();
}
//prijem znaku a tvorba pole znaku pro vykreslovani
static void prijem(void)
{
	while(UCSR0A & (1<<RXC0))
	{
		unsigned char pom_radek, prijaty_znak;
		prijaty_znak = UDR0;
		//kdyz je pole znaku plne, vynuluje se
		if(cislo_akt_znaku == (radku_textu*znaku_na_radek))
		{
			short int i = celkem_znaku;
			char *ukazatel;
			ukazatel = &pole_znaku[0];
			while(i--)
			{
				*ukazatel++ = 0x0;
			};
			cislo_akt_znaku = 0x0;
		}
		switch ( prijaty_znak )
		{
			//osetreni Backspace
			case  8:
				if(cislo_akt_znaku)
				{
					pole_znaku[cislo_akt_znaku] = 0x0;
					pole_znaku[--cislo_akt_znaku] = 0x0;
				}
			break;
			//osetreni Enter
			case  13:
				pom_radek = cislo_akt_znaku / znaku_na_radek;
				pole_znaku[cislo_akt_znaku] = 0x0;
				cislo_akt_znaku = pom_radek*znaku_na_radek +
znaku_na_radek;
			break;
			//pokud neni prijaty znak Backspace ani Enter,
			//tak se prijaty znak zapise do pole znaku,
			//inkrementuje se cislo aktualniho znaku
			default: pole_znaku[cislo_akt_znaku++] = prijaty_znak;
		}
	}
}
```

```
//vykreslovani obrazu
void kresli_obraz()
{
	unsigned char i;
	char * uk1;
	const char * uk2;
	//naplneni pole znaku pro vykresleni uvodni obrazovky
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-20)],&str1[0]);
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-19)],&str2[0]);
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-18)],&str3[0]);
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-15)],&str4[0]);
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-14)],&str5[0]);
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-13)],&str6[0]);
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-12)],&str7[0]);
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-10)],&str8[0]);
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-9)],&str9[0]);
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-6)],&str10[0]);
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-4)],&str11[0]);
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-3)],&str12[0]);
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-2)],&str13[0]);
	strcpy_P(&pole_znaku[znaku_na_radek*(radku_textu-1)],&str14[0]);
	//cislo_akt_znaku se nastavi na konec pole znaku, aby pri
	//prijmu znaku pres USART doslo k vymazani obrazovky,
	//prijate znaky se zacnou vykreslovat od leveho
	//horniho rohu obrazovky
	cislo_akt_znaku = (radku_textu*znaku_na_radek);
	for(;;)
	{
		sleep_mode();
				//kdyz je prijat znak, skok do funkce prijem
		if (UCSR0A & (1<<RXC0))
		{
			prijem();
		}
		//v oblasti aktivniho videa probiha vykreslovani obrazu
		//po dokonceni HS dojde k vykresleni jednoho radku obrazu
		if(povol_vykreslovani)
		{
			//do ukazatele 1 se ulozi adresa prvniho znaku
			//aktualniho radku textu v pole_znaku
			uk1 = &pole_znaku[akt_radek_textu * znaku_na_radek];
			//do ukazatele 2 se ulozi adresa aktualniho radku
			//znaku v matici znaku
			uk2 = &znak[0][akt_radek_znaku >> 1];
			//vykresleni jednoho VGA radku
			i = znaku_na_radek;
			while(i--)
			{
				zapni_vykreslovani;
				//uk1 se kazdym pruchodem cyklu inkrementuje
				//a prechazi tak na dalsi znak na radku
				//kazdy radek matice znaku se kresli dvakrat
				SPDR = pgm_read_byte(uk2 + (*
uk1++)*vyska_znaku/2);
			}
			//prodleva pro vykresleni posledniho znaku,
			//nez se vypne vykreslovani
			NOP;
			NOP;
			NOP;
			NOP;
```

```
                    NOP;
                    NOP;
                    NOP;
                    NOP;
                    NOP;
                    NOP;
                    NOP;
                    NOP;
                    NOP;
                    NOP;
                    NOP;
                    vypni_vykreslovani;
              }
        }
}
int main(void)
{
        ini();
        kresli_obraz();
}
```

6.2 Vysvětlení funkce programu

Na začátku programu jsou uvedeny deklarace a definice, jak je běžné. Za zmínku stojí část, kde jsou definovány řetězce pro vykreslení úvodní obrazovky. Tyto jsou uloženy v programové paměti, aby nezaplňovaly operační paměť.

Následuje synchronizační rutina, která startuje s přerušením od interního čítače 0. Synchronizace je vysvětlena v kapitole 4.1. Navíc je zde to, že při dosažení konce obrazovky se kromě proměnné pro počítání obrazových řádků nulují i proměnné pro počítání řádků textu a aktuálního vykreslovaného řádku znaku. Dále se v oblasti aktivního videa povolí vykreslování a na začátku každého řádku textu se vynuluje proměnná pro počítání řádků znaku. V řádcích obrazu, kde nezačíná nový řádek textu, se musí provést několik instrukcí „No Operation“, které zaberou stejně strojového času, jako nulování proměnné pro počítání řádků znaku, aby vykreslování obrazu začínalo na všech řádcích stejně.

Poté následuje funkce pro inicializaci a nastavení. Zde je povoleno použití jednotek SPI a USART a nastaveny jejich parametry, vše je zřejmé z kódu. Dále je provedeno nastavení idle režimu, nastavení směrového registru portu D pro synchronizaci a nastavení časovače 0 pro spouštění synchronizace. Nakonec je povoleno globalní přerušení.

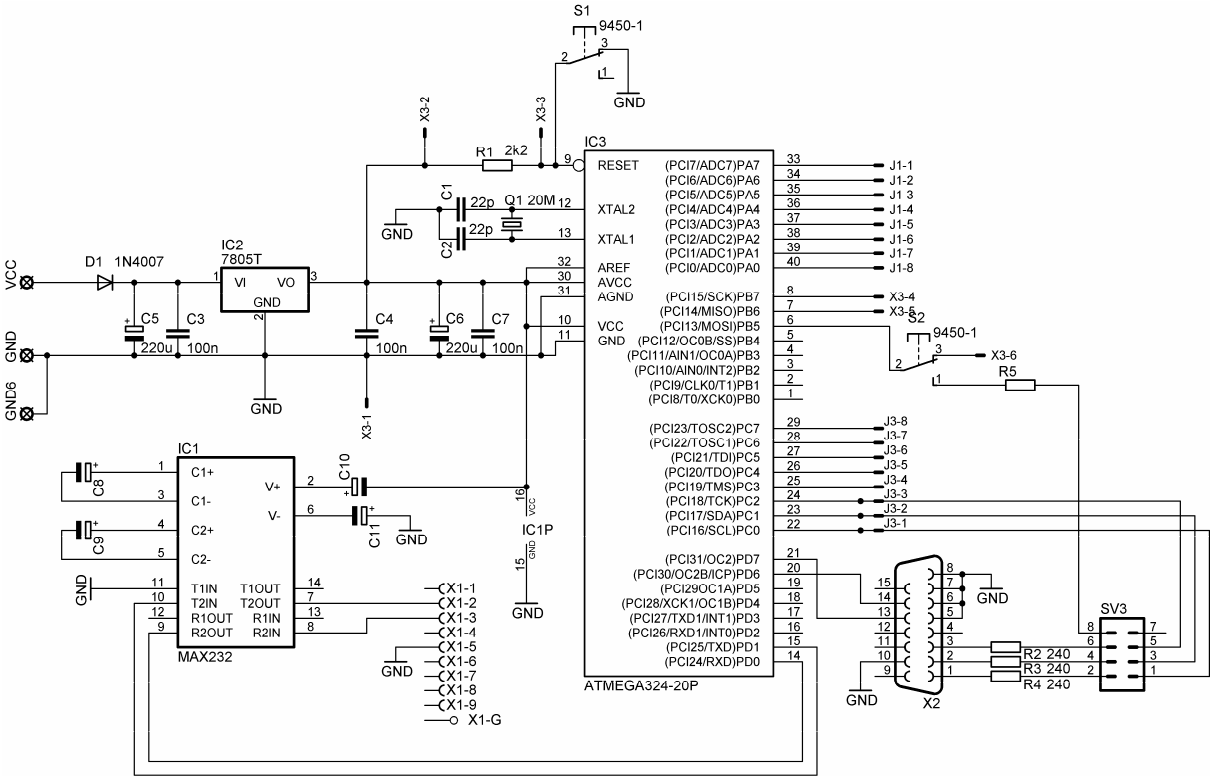
Program pracuje tak, že na začátku každého obrazového řádku je provedena synchronizace a když aktuální obrazový řádek patří do oblasti aktivního videa je, povoleno vykreslování. Ve funkci kresli obraz je nekonečná smyčka, která čeká na povolení vykreslování. Jakmile je vykreslování povoleno, začne vykreslování obrazového řádku. To se děje pomocí cyklu while, kde je zapnuto vykreslování a pak jsou postupně načítány do registru SPDR řádky matic znaků, tak jak jdou po sobě znaky na řádku textu. Po načtení řádku matice posledního znaku na řádku textu se čeká na jeho vykreslení na obrazovku, to je ošetřeno několika instrukcemi „No Operation“, a potom dojde k vypnutí vykreslování. Takto jsou postupně vykreslovány všechny obrazové řádky.

Na začátku nekonečné smyčky je mikrokontroler uveden do režimu idle, ze kterého je probouzen při povolení vykreslování v synchronizační rutině a nebo při přijetí znaku přes sériové rozhraní. To je potřeba proto, aby vykreslování každého řádku začalo přesně stejnou

dobu po impulsu horizontální synchronizace, aby byl obraz na monitoru stabilní. Kdyby nebyl mikrokontroler uváděn do režimu idle, obraz by se na monitoru chvěl. Drobné zachvění po dobu trvání jednoho obrazového řádku se na monitoru vždy objeví při příjmu znaku přes USART, to je způsobeno právě tím, že při kvůli obsluze příjmu znaku se daný obrazový řádek začne vykreslovat se zpožděním.

Na začátku programu jsou do pole znaků nakopírovány řetězce, z nichž je tvořena úvodní obrazovka. Když dojde k příjmu znaku přes sériové rozhraní, skočí se do funkce, která zajišťuje příjem znaků a tvorbu pole znaků pro vykreslování. Činnost této funkce je objasněna na konci kapitoly 5.

7 Přípravek zobrazovacího zařízení



Obr. 10: Schéma zapojení přípravku zobrazovacího zařízení

Na Obr. 10 je zakresleno schéma zapojení přípravku zobrazovacího zařízení, které jsem zkonstruoval jako výsledek této bakalářské práce.

Deska přípravku je osazena mikrokontrolerem Atmel ATmega324P v provedení DIL 40. Mikrokontroler je kvůli snadnější manipulaci a možnosti jednoduché výměny v případě poškození osazen v patici. K mikrokontroleru je připojen krystal o kmitočtu 20 MHz pro generování hodinového taktu.

Dále je na desce osazen integrovaný obvod MAX232, který pracuje jako převodník úrovní pro sériovou linku.

Napájení je řešeno použitím stabilizátoru napětí typu 7805 v klasickém zapojení, který poskytuje napětí 5 V pro mikrokontroler a převodník. Deska je určena pro napájení napětím 9 až 12 V, je jí možné napájet ze síťového adaptéru nebo z 9 V baterie. Na vstupu napájecího přívodu je osazena dioda, která chrání komponenty před poškozením při přepólování napájecího napětí. Napájecí napětí je filtrováno dvěma elektrolytickými kondenzátory 220 μF / 16 V a blokováno keramickými kondenzátory 100 nF umístěnými v blízkosti stabilizátoru a napájecího vývodu mikrokontroleru. Spotřeba celého zařízení je asi 65 mA.

Přípravek dále obsahuje třířadý patnáctipinový konektor CANON – F , který je určen pro připojení monitoru. Na piny 13 a 14 tohoto konektoru jsou přivedeny signály pro horizontální a vertikální synchronizaci obrazu. Barevné vstupy 1, 2 a 3 jsou přes rezistory

240 Ω přivedeny na konektory typ RM s roztečí 2,54 mm (lámací lišta). Vedle těchto konektorů jsou přivedeny na další konektory stejného typu signály z pinů 0, 1 a 2 portu C a signál z pinu MOSI. Takové řešení umožňuje pomocí zkratovacích propojek (jumperů) vybrat barevný režim pro generování grafických obrazců nebo monochromatický režim pro vykreslování textu z pinu MOSI.

Port A a port C mají všechny piny vyvedeny také na konektory RM pro umožnění připojit se na tyto porty při budoucí práci s přípravkem.

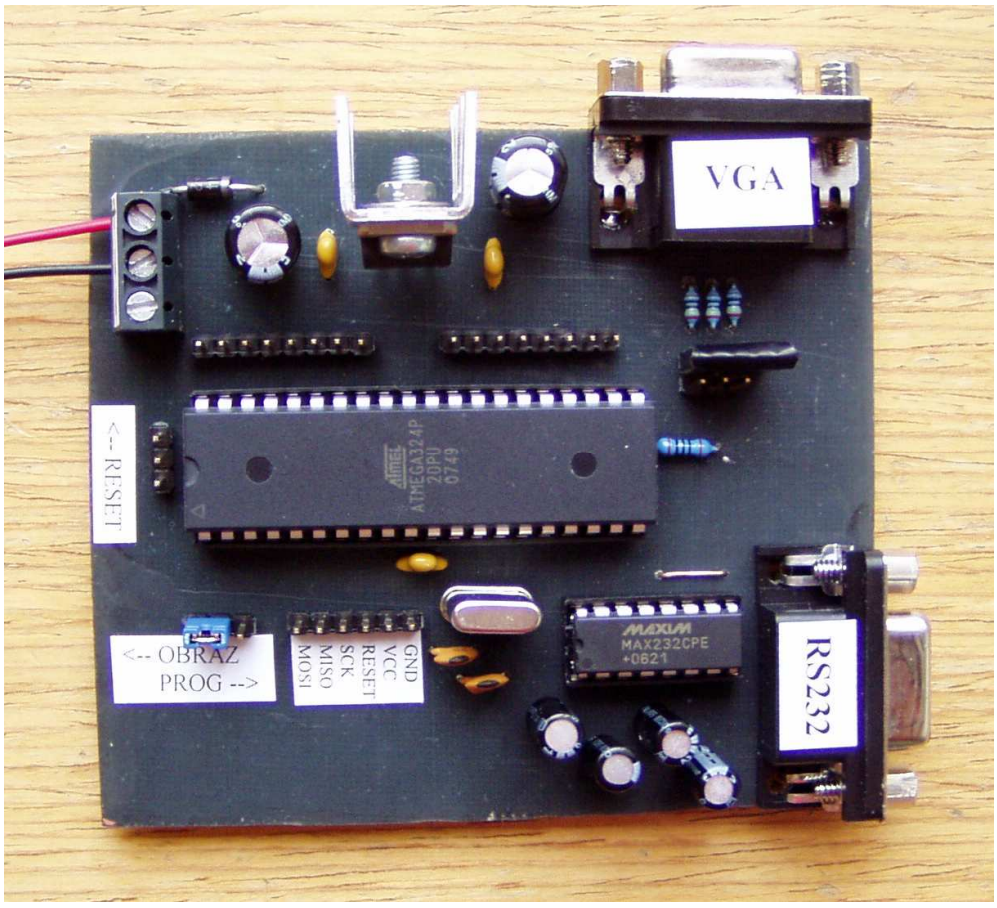
Na konektoy typu RM je vyvedeno i rozhraní SPI pro programování mikrokontroleru. Na pinu MOSI je umístěn přepínač pro zvolení režimu programování nebo režimu vykreslování.

Dále je osazen přepínač Reset, který umožňuje ručně resetovat mikrokontroler.

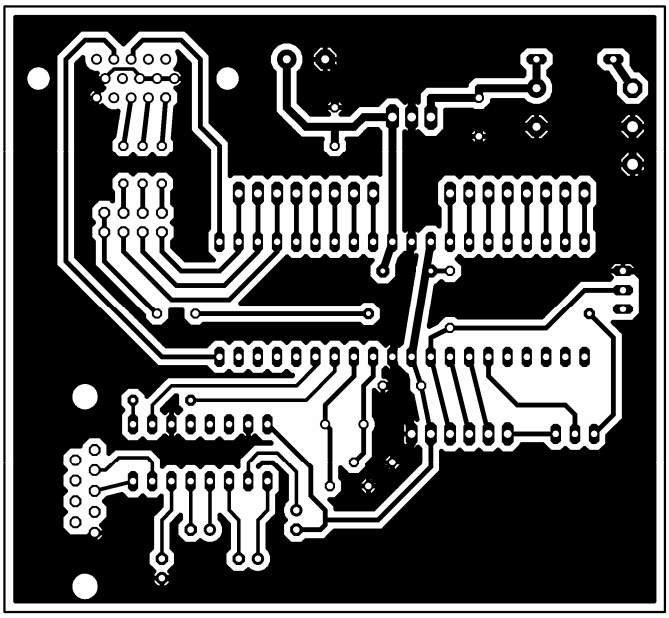
Na desce přípravku je osazen ještě devítipinový konektor CANON – F, který je připojen k převodníku MAX232 a je určen k připojení sériového kabelu pro příjem zobrazovaných dat.

Hodnoty všech součástek jsou patrné ze schématu.

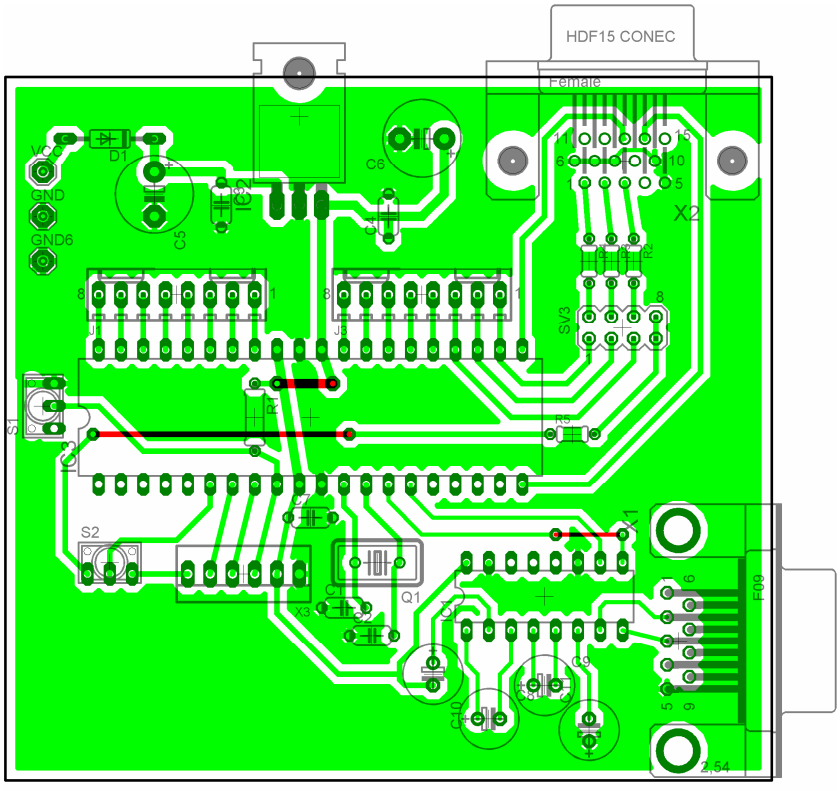
Fotografie přípravku zobrazovacího zařízení je na Obr. 11. Klišé desky plošných spojů je na Obr. 12 a plán osazení součástek na Obr. 13. Klišé plošných spojů v rozlišení 600 dpi je obsaženo na přiloženém CD. Deska přípravku je jednostranná, obsahuje tři drátové propojky. Ty jsou na osazovacím plánu na Obr. 13 znázorněny červenými čarami.



Obr. 11: Pohled na přípravek zobrazovacího zařízení



Obr. 12: Klišé plošných spojů



Obr. 13: Plán osazení součástek

8 Programování mikrokontroleru

8.1 Software

K vytváření programového vybavení pro účely této práce jsem použil aplikaci AVR Studio 4. Tato aplikace pochází přímo od výrobce mikrokontrolerů AVR, firmy Atmel. Na internetových stránkách výrobce http://atmel.com/dyn/products/tools_card.asp?tool_id=2725 je v současné době (červen 2008) k dispozici ke stažení verze 4.14. Tato aplikace je poskytována zdarma, ke stažení je požadována pouze bezplatná registrace.

Tato aplikace umožňuje programování v jazyce symbolických adres nebo v jazyce C. Pro programování v jazyce C je vyžadován externí kompilátor. Jako kompilátor používám WinAVR. Tento programový balík je k dispozici ke stažení na http://winavr.sourceforge.net/ . Program je poskytován jako opensource. Tento kompilátor se integruje do AVR studia a nevyžaduje žádné složité nastavení, pro správnou funkci je pouze zapotřebí instalovat WinAVR před instalací AVR Studia, aby došlo k automatické integraci kompilátoru.

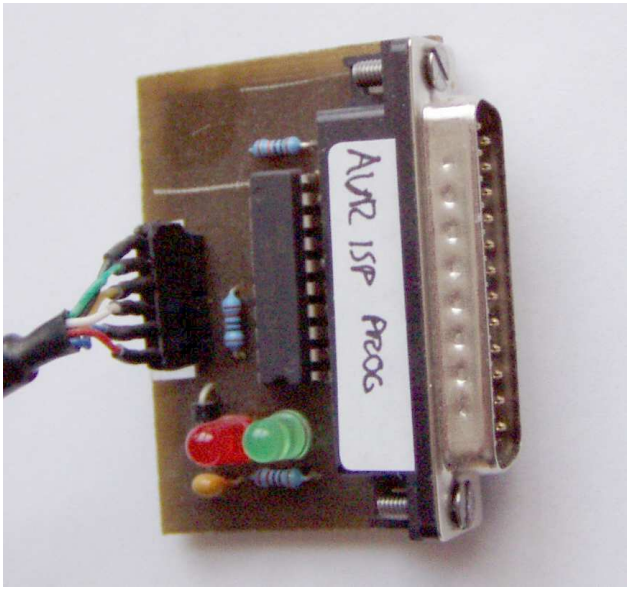
Pro nahrávání zkompilovaného hex souboru do mikrokontroleru používám program PonyProg. Tento software umožňuje programování široké škály různých zařízení, např. mikrokontrolery AVR, PIC, paměti EEPROM od různých výrobců atd. Program je možné stáhnout ze stránek http://www.lancos.com/ppwin95.html , je poskytován jako opensource.

Pro posílání zobrazovaných znaků do mikrokontroleru používám program Terminal. Tento program je používán i v laboratořích UREL při výuce předmětu Mikroprocesorová technika. Program jsem získal ze stránek http://www.rs232.hw.cz . Stránky autora uvedené v nápovědě k programu již nejsou v současné době funkční.

Veškeré programy, které jsem použil při řešení této bakalářské práce, jsou obsaženy na přiloženém CD v adresáři INSTAL.

8.2 Hardware

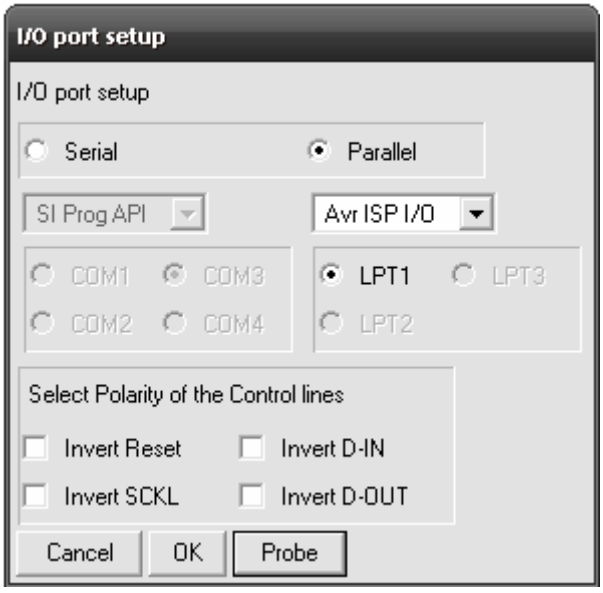
K nahrávání programu do mikrokontroleru je zapotřebí tzv. programátor. Pro práci na tomto projektu používám jednoduchý programátor AVR ISP prog. Návod na jeho konstrukci lze najít na stránkách programu PonyProg, [7]. Tento programátor se připojuje na paralelní port osobního počítače a programuje mikrokontroler přes rozhraní SPI. Na přípravku zobrazovacího zařízení je toto rozhraní také vyvedeno, takže pro programování stačí propojit programátor a přípravek kabelem a není nutné mikrokontroler vyjímat z desky.



Obr. 14: Programátor AVR ISP prog

8.3 Práce s programem PonyProg

Program je nejprve nutné nastavit, aby komunikoval s mikrokontrolerem. Je zapotřebí vybrat komunikační port a typ zařízení, které chceme programovat. K programování mikrokontroleru AVR přes AVR ISP prog musíme v menu Setup -> Interface Setup nastavit programování přes paralelní port a AVR ISP I/O, jak je patrné z Obr. 15. V menu Device je ještě potřeba nastavit typ zařízení „AVR micro“ a vybrat konkrétní typ mikrokontroleru, v našem případě tedy ATmega324. Komunikaci vyzkoušíme tlačítkem Probe, pokud se vypíše hláška „Test OK, je všechno v pořádku. Poté ještě provedeme kalibraci (menu Setup -> Calibration) a máme program nastaven pro práci s mikrokontrolerem.



Obr. 15: Nastavení programu PonyProg

Když máme správně nastaven program, můžeme přistoupit k nastavení mikrokontroleru. Program PonyProg umožňuje nastavování konfiguračních a bezpečnostních zámků mikrokontroleru. To je velmi důležité, protože procesory Atmel AVR jsou dodávány přednastavené tak, aby používaly interní oscilátor, který pracuje na frekvenci 1 MHz. Nově zakoupený mikrokontroler tedy nepoužívá krystalový oscilátor. Toto je třeba změnit. Otevřeme položku z menu Command -> Security and Configuration Bits a nastavíme podle Obr 16. Důležité pro volbu pracovního kmitočtu jsou položky CKSEL 0 až 3. Je také vhodné zakázat funkce, které nepotřebujeme, např. rozhraní JTAG, které když je povoleno, tak blokuje některé piny na portu C. Bližší informace o nastavení zámků lze najít v datasheetu mikrokontroleru ATmega324P, [6].

Nyní už máme nastaveno všechno potřebné pro práci s mikrokontrolerem. Pro nahrání hex souboru do mikrokontroleru jej otevřeme (menu File -> Open Device File) a příkazem z menu Command -> Write All nebo klávesovou zkratkou <Ctrl – W> zapíšeme do mikrokontroleru.

Pomocí programu můžeme také přečíst z mikrokontroleru hex soubor, který je v něm uložený, to provedeme příkazem z menu Command -> Read All, klávesová zkratka <Ctrl – R>.

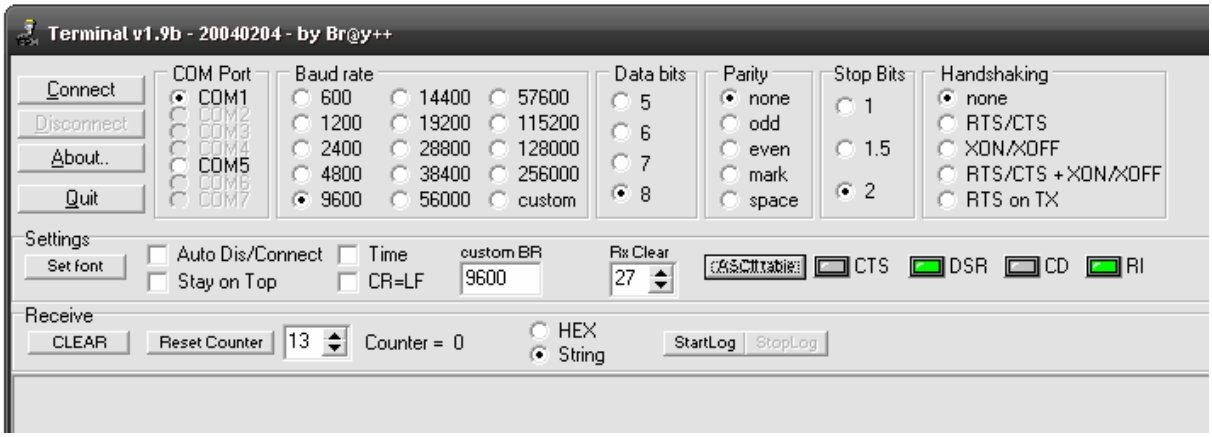
Obr. 16: nastavení bezpečnostních a konfiguračních zámků v programu Pony Prog

8.4 Práce s programem Terminal

Program Terminal je jednoduchý software který slouží pro komunikaci přes sériové rozhraní. Nabízí veškeré nastavení, které je pro komunikaci nutné.

Pro odesílání dat do mikrokontroleru je třeba správně zvolit sériový port, na kterém je připojen přípravek zobrazovacího zařízení, nastavit přenosovou rychlost na 9600 b/s, počet datových bitů na 8, žádnou paritu a žádný handshaking. Nastavení je patrné z obrázku 17. Poté stačí kliknout na tlačítko Connect a začít psát. Vypsané znaky se začnou vykreslovat na monitoru připojeném k přípravku zobrazovacího zařízení.

Pro komunikaci s přípravkem zobrazovacího zařízení jsem zkoušel také program Hyperterminal, který je součástí operačního systému Windows. Odesílání znaků do zobrazovacího zařízení bylo také funkční, ale program Terminal nabízí mnohem lepší uživatelský komfort a také větší spolehlivost, než program Hyperterminal.



Obr. 17: Okno programu Terminal s nastavením

9 Závěr

V této práci bylo objasněno, jak pracuje VGA rozhraní a bylo vysvětleno, jakým způsobem je možné vytvářet obraz v mikrokontroleru.

Hlavním úkolem práce bylo vytvořit funkční přípravek zobrazovacího zařízení.

V rámci práce na projektu byl navržen a vyroben funkční výrobek zobrazovacího zařízení, osazený mikrokontrolerem Atmel ATmega324P.

Pro tento výrobek byly vytvořeny programy, které umožňují vytváření a zobrazování jednoduchých barevných obrazců nebo textu v rozlišení 25 znaků na řádek a 20 řádků textu, celkem 500 znaků současně zobrazených na obrazovce.

Dílčím úkolem bylo navrhnout způsob získávání dat pro zobrazení v mikrokontroleru.

Pro získávání zobrazovaných dat bylo použito sériové rozhraní mikrokontroleru USART, data pro zobrazení jsou mikrokontrolerem přijímána přes toto rozhraní a zobrazována na monitoru.

Při práci na projektu byla vyzkoušena synchronizace obrazu, vytváření grafiky a textu na monitoru a komunikace zobrazovacího zařízení s osobním počítačem přes sériové rozhraní. Výsledkem je přípravek zobrazovacího zařízení, který bude spolu s programem, umožňujícím zobrazovat na monitoru text uložený v paměti nebo přijímaný přes sériové rozhraní, předveden při obhajobě této bakalářské práce.

10 Seznam literatury

[1] PINOUTS.RU Team. *VGA pinout and signals* [online]. [cit. 2008-06-05]. Dostupný z WWW: http://pinouts.ru/Video/VGA15_pinout.shtml .

[2] SEDLÁČEK, Petr. *Textová VGA grafická karta* [online]. Plzeň 2006 [cit. 2008-06-05]. Dostupný z WWW: http://www.volny.cz/ekrakonos/vga1/cz_vga1.htm .

[3] FRÝZA, Tomáš, FEDRA, Zbyněk, ŠEBESTA, Jiří. *Mikroprocesorová technika Laboratorní cvičení* [online]. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2007 [cit. 2008-06-05]. Dostupný z WWW: https://krel.feec.vutbr.cz/VYUKA/B_EST/prezencni/BMPT/laboratore/skripta/bmpt_laboratore_071129.pdf .

[4] IBRAGIMOV, Maxim Rafikovich. *Simple VGA/Video Adapter* [online]. Togliatti 2005 [cit. 2008-06-05] Dostupný z WWW: http://www.serasidis.gr/circuits/AVR_VGA/avr_vga.htm.

[5] MATOUŠEK, David. *Práce s mikrokontroléry ATMEL AVR ATmega16.* Praha: BEN – technická literatura, 2006. 320 s. ISBN 80-7300-174-8.

[6] ATMEL Corporation. *8-bit AVR microcontroller ATmega164P/ ATmega324P/ ATmega644P* [online]. 2007 [cit. 2008-06-05].Dostupný z WWW: http://atmel.com/dyn/resources/prod_documents/doc8011.pdf

[7] LANCONELLI, Claudio. *PonyProg – Serial Device Programmer* [online]. 2007 [cit. 2008-06-05]. Dostupný z WWW: http://www.lancos.com/prog.html#hardware .