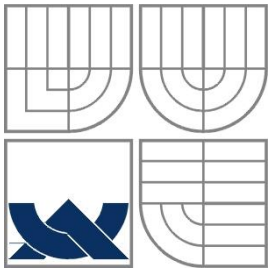
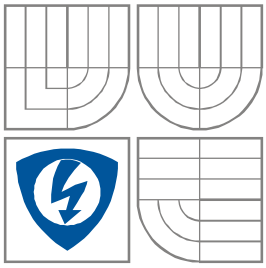


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

KNIHOVNA GRAFICKÝCH PRVKŮ PRO MIKROKONTROLERY ATMEL AVR A GRAFICKÉ DISPLEJE

LIBRARY FOR GRAPHIC OBJECTS FOR MICROCONTROLLERS ATMEL AVR AND GRAPHIC DISPLAYS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Miroslav Skopal
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:	Miroslav Skopal
Bytem:	Válkova 180, Červenka, 784 01
Narozen/a (datum a místo):	26. srpna 1983 ve Šternberku

(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:	Knihovna grafických prvků pro mikrokontrolery Atmel AVR a grafické displeje
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.

* hodící se zaškrtněte

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

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

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma knihovna grafických prvků pro mikrokontrolery Atmel AVR a grafické displeje 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

Anotace

Tato bakalářská práce je zaměřen na problematiku ovládání grafických LCD displejů pomocí mikroprocesorů řady AVR. Podrobně je zde popsán řadič od firmy Toshiba T6963C. Výsledkem této práce je modulární knihovna napsaná v jazyku C pro mikrokontrolery Atmel AVR, nabízející sadu funkcí pro kreslení jednoduchých grafických objektů a ovládacích prvků. Knihovna je postavena jako vícevrstvá, čímž umožňuje použití různých řadičů bez výraznějšího zásahu do výsledného programu.

Abstract

This bachelor's thesis is concerned about the graphic LCD display control by the Atmel´s AVR microprocessors. Toshiba T6963 controller is described in detail. The overall result of my bachelor's thesis is the modular library written in C language (GCC). It provides a set of functions for simple graphic objects and controls drawing. The library is designed as a multi - layered one. It supports the usage of different LCD display controllers without any extensive program code modification.

Klíčová slova

LCD displej, Atmel AVR, mikroprocesor, řadič T6963C, M2606F, knihovna, GCC, AVR studio, WinAVR, grafika, Bresenhamův algoritmus, rasterizace, grafický režim, ovládací prvky, bitmap font

Key words

LCD display, Atmel AVR, microprocessor, controller T6963C, M2606F, library, GCC, AVR studio, WinAVR, graphics, Bresenham algorithms, raster graphics, graphics mode, controls, bitmap font

Obsah

SKOPAL, M. Knihovna grafických prvků pro mikrokontrolery Atmel AVR a grafické displeje. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 47 s.
Vedoucí bakalářské práce Ing. Zbyněk Fedra, Ph.D.

Seznam tabulek

Seznam obrázků

1. Úvod

Cílem této práce je vytvoření knihovny nabízející jednoduché rozhraní pro ovládání grafických LCD displejů prostřednictvím mikrokontroleru firmy Atmel z řady AVR. Jedná se tedy především o práci zaměřenou na softwarovou část řešení této problematiky. Informace o hardwarovém zapojení LCD displejů zde budou zmíněny pouze okrajově v rozsahu nezbytně nutném pro realizaci testovacího obvodu pro odzkoušení a demonstraci výsledné knihovny.

Knihovnu by mělo být možné využívat u všech mikrokontrolerů řady AVR, které disponují dostatečným počtem vstupně výstupních portů a dostatečným paměťovým prostorem jak pro samotný program ve FLASH paměti nebo stavové informace uložené v RAM během provozu zařízení.

Pro vytvoření této knihovny bylo využito nástroje AVR Studio ve verzi 4.13 od společnosti Atmel [4] a doplňkového modulu WinAVR, který umožňuje zápis programu pro mikrokontrolery v jazyku C. WinAVR využívá pro získání výsledného strojového kódu The GNU Compiller Collection (GCC). Velkou výhodou využití toho překladače je nesrovnatelně větší přehlednost zdrojového kódu a z toho vyplívající jednodušší vývoj a následná správa hotových programů. Za zmínku určitě také stojí propracovaný optimalizační nástroj, který samostatně volí nejefektivnější varianty překladu programu do strojového kódu.

2. Hardware

2.1. Mikrokontrolery řady AVR

Řada AVR označuje rodinu 8 bitových RISC (*Reduced Instruction Set Computer)* mikroprocesorů s maximální pracovní frekvencí 16MHz (u některých modelů i 20MHz). Díky relativně vysoké taktovací frekvenci a sadě instrukcí, z nichž je většina vykonávána během jednoho hodinového taktu, jsou mikroprocesory řady AVR jedny z nejvýkonnějších mikroprocesorů pro obecné použití. Určitě neméně zajímavou informací je jejich cena, která se pohybuje řadově v desítkách korun pro kusové odběry. Mikrokontrolery Atmel jsou například k dostání u společnosti GM Electronic, kde jsem zakoupil i mikrokontroler ATMega16 (cena 82 Kč), na kterém byla testována vytvořená knihovna.

2.2. Mikrokontroler ATMega16

Pro vývoj a testování knihovny jsem zvolil mikroprocesor s označením ATMega16 pro jeho dostatečnou programovou paměť FLASH a operační paměť RAM. Pro samotné řízení LCD displejů se využívá velice malá část výbavy, kterou tento mikroprocesor disponuje. Tabulka 1 popisuje některé základní vlastnosti důležité nebo související s funkčností vytvořené knihovny.

Pracovní frekvence	0-16MHz
Programová paměť FLASH	16 kB
Operační paměť	1 kB
Porty	32x programovatelné I/O
Čítač/Časovač	2x 8bit, 1x16bit

tabulka 1 - vybrané vlastnosti ATMega16

Pro konkrétní realizaci grafické knihovny budeme využívat pouze 13 vstupně výstupních portů a to 8 pro datovou sběrnici a 5 pro řídící signály. Dále může být využito časovačů na zpomalení a načasování komunikace s LCD displejem. Programová a operační paměť je nezbytná pro chod všech programů, avšak je důležité upozornit, že grafická knihovna si alokuje určitou oblast RAM paměti, které je využita pro uchování interních stavových proměnných. Tato část paměti je trvale osazena a nelze ji dále využívat!

Nejdůležitějším zdrojem informací potřebných pro prácí s tímto mikroprocesorem je jeho katalogový list dostupný ke stažení na stránkách firmy Atmel [7].

2.3. LCD Displeje

Slovo LCD je zkratka [8] z anglických slov Liquid crystal display, která by se dala přeložit jako panel z tekutých krystalů. Technologie LCD displejů je již v dnešní době na poměrně vysoké úrovni a můžeme se tak setkat v běžné praxi jak s monochromatickými LCD displeji tak i s vysoce kvalitními barevnými LCD panely. Tato práce je zaměřena pouze na monochromatické LCD displeje, které se hojně využívají v různých elektrotechnických zařízeních, kde zprostředkovávají komunikaci s obsluhou. Tyto monochromatické (Označení černobílé není příliš namístě z důvodů, že většina těchto displejů nepoužívá k zobrazení černé a bílé barvy, ale jiné kontrastní kombinace. Nejčastěji se můžete setkat s kombinací černá a zelená.) displeje se dají dělit dále na tři základní skupiny.

2.3.1. Numerické displeje

S numerickými displeji se nejčastěji setkáváme například u většiny kalkulaček a hodinek. Jedná se o základní displej zobrazující pouze cifry 0 až 9 pomocí sedmi-segmentového zobrazení. Displeje mohou být vybaveny různým počtem zobrazovaných cifer nebo popřípadě doplněny jinými znaky jako je plus nebo mínus. Pro tyto displeje je typické, že jsou schopny zobrazit pouze předem definované znaky, které jsou dány tvary jednotlivých segmentů displeje. Díky dnes již nízkým výrobním nákladům si jednotliví výrobci nechávají vyrábět speciální LCD displeje pro své zařízení s množstvím dalších speciálních znaků. Z tohoto důvodu se s čistě numerickým displejem setkáte jen velmi zřídka.

2.3.2. Alfanumerické displeje

Alfanumerické displeje jsou dnes v elektrotechnice jedny z nejrozšířenějších typů LCD displejů. Jedná se o maticové LCD displeje, které odpovídající znaky zobrazují pomocí matice bodů (nejčastěji 5x8 nebo 8x8 bodů). Toto uspořádání je

náročnější na hardware obsluhy LCD displeje oproti numerickým displejům, ale přináší obrovské výhody a to hlavně, že již displej není omezen výrobně danými zobrazovanými symboly. Pro ovládání displeje se většinou používají tzv. řadiče. Tyto řadiče jsou integrovány přímo na modulu LCD panelu a jsou ovládány pomocí řídicí a datové sběrnice. Nepsaným standardem pro ovládání alfanumerických displejů se stal řadič od firmy Hitachi HD44780. Více informací o tomto řadiči můžete najít například na internetových stránkách.

2.3.3. Grafické displeje

Grafické LCD displeje se na první pohled výrazně neliší od displejů alfanumerických. Vyrábějí se ve stejném kontrastním provedení, ovládají se také pomocí řídicí a datové sběrnice a zobrazují znaky pomocí matice bodů. I přes tuto podobnost se dají tyto dva typy displejů od sebe rozpoznat při pouhém pohledu. V prvním případě není potřeba ani příliš zkušeného oka a stačí se podívat na zadní stranu LCD modulu, kde většinou největší integrovaný obvod je právě řadič displeje. Když už znáte označení řadiče, není žádný problém vyhledat si příslušný katalogový list na internetu, kde je přímo napsáno, pro jaký typ LCD displeje je řadič určen. Další možností, jak rozpoznat typ displeje, je pozorně se podívat na zobrazovací část (obrazovku) v určitém úhlu, kde v případě alfanumerického displeje uvidíte mřížku vytvořenou mezerami mezi jednotlivými znaky. V případě grafického displeje neuvidíte prakticky nic. Grafický displej tvoří jedna velká matice bodů s rozměry určujícími celkové rozlišení displeje. Nejčastěji se můžete setkat s displeji s horizontálním rozlišením 120, 128, 160, 240 a 320 bodů a vertikálním rozlišením 32, 64, 128 a 240 bodů. Stejně jako alfanumerický displej je grafický displej řízen svým řadičem. Řadiče pro grafické displeje jsou podstatně výkonnější než alfanumerické řadiče a nejčastěji se můžete setkat s obvody řadičů s označením T6963, SED1520, SED1334 a jiné. Každý řadič má dáno maximální rozlišení displeje, které je schopen obsluhovat. V této práci je pro demonstraci použit LCD modul s označením MG2406F s rozlišením 240 x 64 bodů s integrovaným radičem od firmy Toshiba T6963C [3]. Tento radič je níže podrobně popsán a veškeré zde uvedené informace a postupy jsou vztaženy přímo k tomuto řadiči.

2.3.4. Na co si dávat pozor při výběru LCD modulu

Většina dnes prodávaných LCD modulů je již vybavena integrovaným řadičem, ale může se i objevit LCD modul bez řadiče. Použití takového LCD modulu přináší velké množství problémů. Samozřejmě lze ovládat i LCD modul bez řadiče přímo pomocí mikroprocesoru, ale je dobré si uvědomit, že režie obnovy všech zobrazovaných bodů na displeji plně zaměstná i relativně silný mikroprocesor. Proto je určitě vhodnější vždy použít LCD modul s integrovaným řadičem.

Řadič HD44780 a jiné kompatibilní řadiče umožňují použití 4bitové datové sběrnice. V případě, že používáte mikroprocesor s malým počtem I/O portů můžete tak využít této 4bitové sběrnice namísto 8 bitové, čímž ušetříte 4 I/O porty.

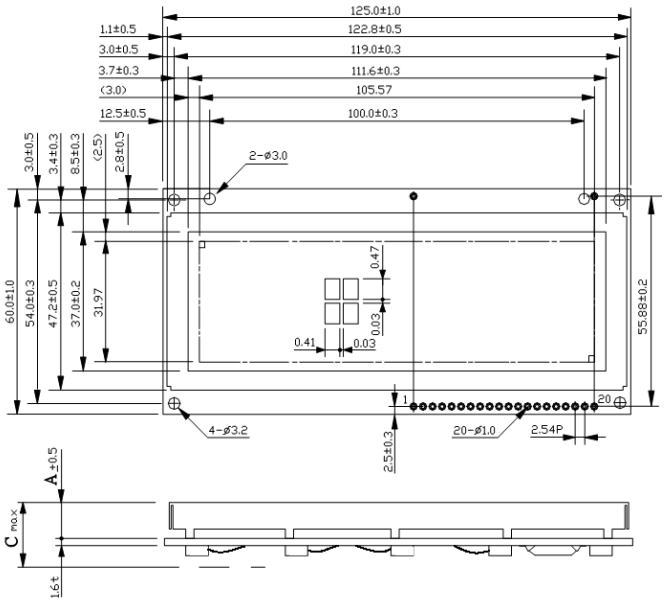
Dalším nepříjemným překvapením může být absence konvertoru napájecího napětí na budící napětí LCD displeje. Jak je známo, displej z tekutých krystalů potřebuje ke své funkci velmi malý proud, ale o to větší napětí. Proto je většina LCD modulů vybavena měničem napájecího napětí na budící napětí (řádově desítky voltů) displeje.

Pokud budete chtít použít LCD modul v prostředí s proměnlivou teplotou, určitě není na škodu, aby LCD modul obsahoval teplotně kompenzovaný zdroj záporného napětí pro řízení kontrastu displeje. Vyhnete se tím nepříjemnému kolísání kontrastu se změnou teploty. Ovšem nezapomínejte, že naprostá většina LCD displejů není schopna pracovat při teplotě bod bodem mrazu, kdy dochází k zamrznutí krystalů a displej tak přestává zobrazovat. V takových podmínkách je nutno použít speciální displeje nebo zajistit vyhřívání LCD displeje ze spodu.

Při pořizování nového LCD displeje je vždy vhodnější volit LCD modul, který používá obecně známé řadiče jako jsou například HD44780 pro alfanumerické displeje nebo T6963c pro grafické displeje. Vyhnete se tak obtížnému a zdlouhavému vyhledávání informací o nějakém „exotickém“ řadiči.

2.4. LCD modul MG2406F

Jak již bylo zmíněno výše, při demonstraci této práce byl použit grafický displej od firmy Everbouquet international Co., LTD. s označením MG2406F [9]. Tento monochromatický displej nabízí rozlišení 240 x 64 bodů pod kontrolou řadiče od firmy Toshiba T6963c. Dále je modul vybaven obvodem od firmy Sanyo LC3564BM (64Kb SRAM paměť, význam paměti viz níže). Na spodní straně displeje se nachází řada dvaceti pájecích plošek, na které je nutno napájet pinovou lištu. Na tyto pájecí plošky jsou vyvedeny vodiče datové a řídicí sběrnice, zdroj záporného napětí pro nastavení kontrastu, anoda a katoda pro napájení LED podsvětlením.

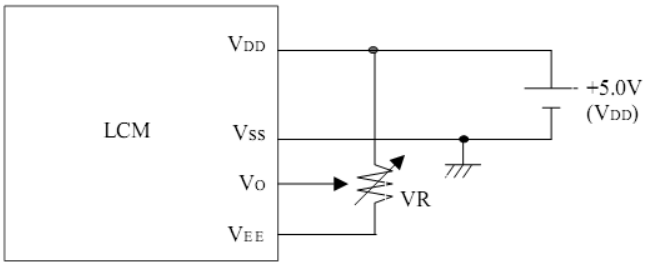


obrázek 1 - LCD modul MG2406F [9]

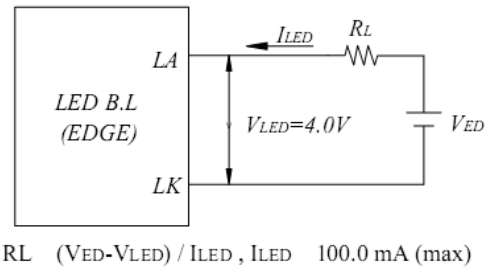
Nejjednodušší způsob, jak připojit LCD modul k mikroprocesoru je tzv. Pin-to-Pin (pin na pin). Jedná se o zapojení, kdy jsou vodiče datové a řídicí sběrnice přímo připojeny na I/O porty mikroprocesoru. Toto zapojení je velice jednoduché a nevyžaduje použití dalších součástek, ale obsadí poměrně velký počet vstupně výstupních portů mikroprocesoru. Následující tabulka a obrázek popisují zapojení pinů LCD modulu.

Číslo pinu	Označení	Funkce
1	V_{SS}	Zem (GND)
2	V_{DD}	Napájecí napětí (+5V)
3	V_O	Řídicí napětí pro kontrast displeje
4	$C/\overline{D}$	Výběr mezi instrukcemi a daty (H: instrukce, L: data)
5	$\overline{RD}$	L: Čtení dat nebo statusu
6	$\overline{WR}$	L: Zápis dat nebo instrukcí
7	DB0	Datová sběrnice (LSB)
8	DB1	Datová sběrnice
9	DB2	Datová sběrnice
10	DB3	Datová sběrnice
11	DB4	Datová sběrnice
12	DB5	Datová sběrnice
13	DB6	Datová sběrnice
14	DB7	Datová sběrnice (MSB)
15	$\overline{CE}$	L: modul aktivní
16	$\overline{RES}$	L: reset modulu
17	V_{EE}	Zdroj záporného napětí pro V_O
18	LC	H: zapíná podsvětlení (nepoužito)
19	LK	Katoda LED podsvětlení (+)
20	LA	Anoda LED podsvětlení (-)

tabulka 2 - zapojení pinů LCD modulu MG2406F [9]



obrázek 2 - řízení kontrastu LCD displeje [9]



obrázek 3 - zapojení podsvětlení LCD displeje [9]

2.5. Řadič Toshiba T6963c

Jedná se o obvod speciálně vyvinutý pro řízení monochromatických grafických LCD displejů střední velikosti. Maximální rozlišení pro tento řadič je dáno jeho pracovní frekvencí 5.5 MHz a velikostí externí paměti RAM, kam se ukládají veškerá zobrazovaná data. Tento řadič je schopen obsluhovat externí paměť RAM až do velikosti 64Kb, která umožňuje uložit až 4 obrazovky o rozlišení 240x64 bodů. Dále tento řadič dokáže pracovat ve třech různých režimech a to v textovém, grafickém a kombinovaném.

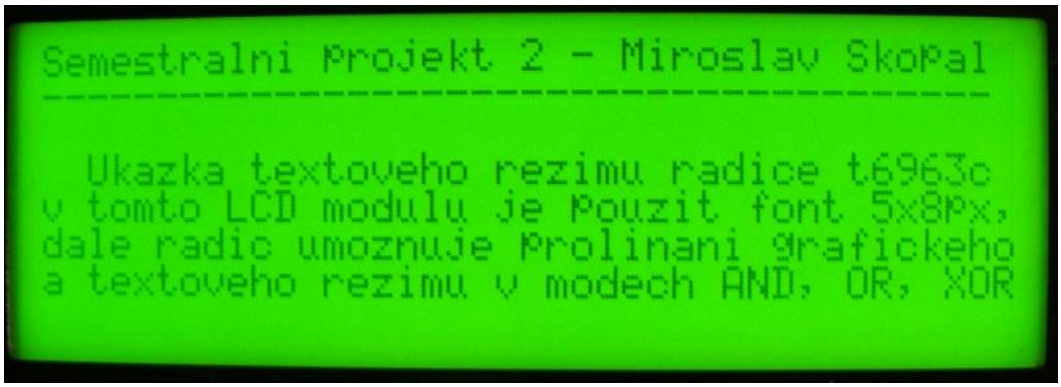
Funkce řadiče spočívá v tom, že periodicky obnovuje všechny body zobrazené na displeji podle obsahu externí paměti RAM. Do této paměti pak pomocí instrukcí zapisujeme příslušná data, která jsou poté prezentována na LCD displeji jako skupina bodů, v případě zápisu dat do části paměti vyhrazené pro grafiku nebo jako odpovídající znak při zápisu do textové části paměti. Rozdělení (přiřazení) externí RAM paměti se obvykle provádí při inicializaci LCD displeje. Ve speciálních případech je možné měnit rozdělení externí RAM paměti za běhu programu. Díky tomuto přerozdělení lze dosáhnout velice rychlých efektů jako je posun obrázků po displeji.

Inicializaci LCD displeje je nutné provádět vždy po jeho připojení k napájecímu napětí. Při neprovedení inicializace se mohou na displeji zobrazovat náhodné znaky nebo jiné náhodné obrazce. Jak již bylo napsáno výše, řadič pouze periodicky zobrazuje obsah externí RAM paměti od nastavené adresy a ve zvoleném rozsahu. Jelikož se jedná o paměť typu RAM, je její obsah po odpojení od napájecího zdroje vždy ztracen a je více něž vhodné ji po opětovném připojení napájení vynulovat (zapsat do celé paměti 0x00).

2.5.1. Textový režim

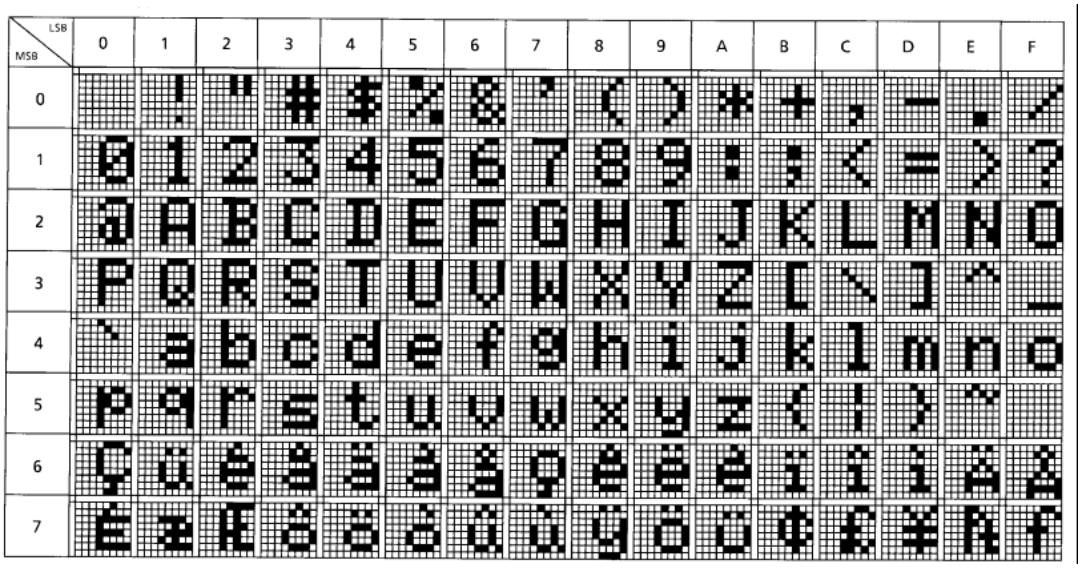
Řadič T6963c obsahuje interní generátor znakové sady o velikosti 128 znaků. Znaková sada začíná prázdným znakem (space) s hodnotou 0 a pokračuje standardně jako ASCI sada znakem „!“ až po znak s kódem 0x7f. Základní znaky abecedy tedy nelze v kódu zapisovat přímo, ale musí se od každého ASCI znaku odečíst hodnota 0x20.

Tento řadič disponuje i dalšími zajímavými funkcemi pro zobrazení textu, jako je například blikající text nebo inverzní zobrazení, ale jelikož je práce především zaměřena na ovládání grafických displejů, zůstaneme pouze u základních funkcí textového režimu.



obrázek 4 - textový režim

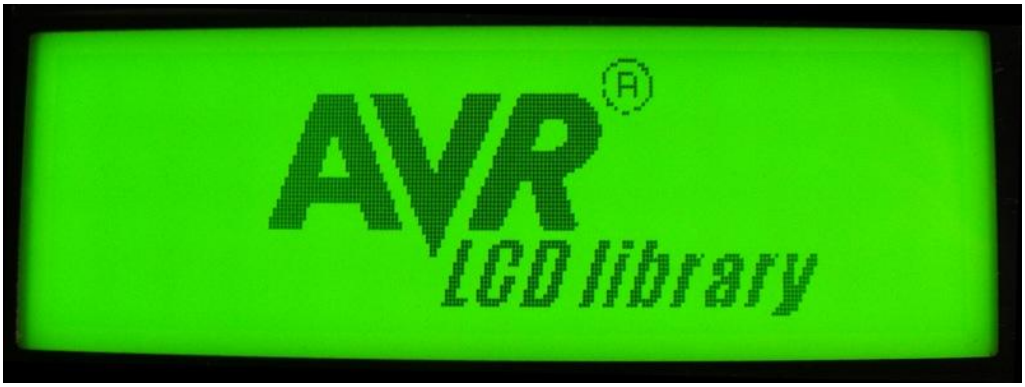
Na obrázku 5 je interní znaková sada řadiče T6963C. Tabulka obsahuje grafickou prezentaci znaku a jeho kód.



obrázek 5 - interní znaková sada [3]

2.5.2. Grafický režim

Oproti textovému režimu, kde byla nejmenší volitelně zobrazitelná jednotka celý znak, grafický režim může zobrazit pouze jediný bod (pixel). Protože se jedná o monochromatický displej pro uchování informace o bodu postačí pouze jeho vertikální a horizontální souřadnice a barva. Souřadnice bodu je dána umístěním (adresou) v externí paměti RAM a barvu pak udává odpovídající bit v 8bitové paměťové buňce (True bod je zobrazen, False bod není zobrazen).



obrázek 6 - grafický režim

2.5.3. Kombinovaný režim

Kombinovaný režim umožní současné zobrazení jak grafické části paměti, tak i textové. Řadič T6963c podporuje tři režimy kombinovaného zobrazení.

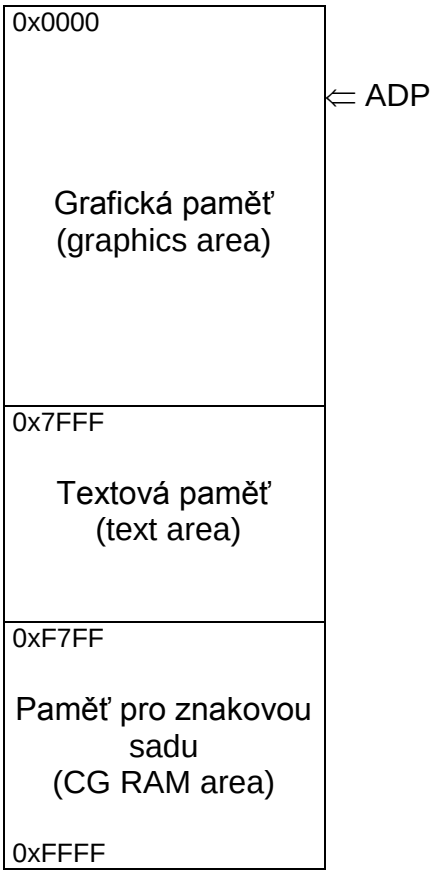
1. AND - výsledný obraz na LCD displeji je složen pouze z bodů z grafické a textové paměti, které jsou vykreslovány na stejné souřadnice displeje.
2. OR – grafická a textová část jsou vykresleny bez jakéhokoliv vzájemného ovlivnění (viz obrázek 7)
3. EXOR – body z grafické paměti se stejnými souřadnicemi jako body z textové paměti jsou vykresleny inverzní barvou

obrázek 7 - kombinovaný režim

Použití kombinovaného zobrazení by bylo vhodné například pro měřící přístroje, kde v jedné části LCD displeje bude zobrazován průběh měřeného signálu a v druhé informace o frekvenci, amplitudě a jiných důležitých vlastnostech signálu.

2.6. Externí paměť RAM

Externí paměť RAM (realizována jako obvod LC3564BM) je k řadiči připojena přes 8bitovou paralelní sběrnici a slouží řadiči jako zdroj dat pro zobrazení na LCD displeji. Jelikož jsou data v paměti uložena jako klasická 8bitová slova, je nutné, aby řadič znal přesné formátovaní dat v paměti, a tak mohl přesně reprodukovat data z paměti na displej. Formátování dat volíme v inicializaci displeje. Na následujícím obrázku je příklad rozdělení externí RAM paměti. Jednotlivé části volíme s ohledem na potřebnou velikost pro daný režim.



obrázek 8 - rozdělění externí RAM paměti

Příklad výpočtu velikosti paměti pro jednu grafickou stránku s rozlišením 240 x 64 bodů

Počet sloupců

240 / 6 = 40
*(1 bod je uložen jako 1 bit v 8bitovém slově, proto jeden byte obsahuje informace o osmi po sobě jdoucích bodech se stejnou adresou, ale řadič T6963C používá režim, kdy jsou poslední 2 (MSB) bity ignorovány**, tedy každý 1 byte (8 bitů) paměti obsahuje informaci pouze o 6 bodech**)*

Počet řádků odpovídá plnému vertikálnímu rozlišení, tedy 64

Výsledná velikost paměti pro jednu grafickou stránku

40 * 64 = 2560 bytů

2.7. Komunikace s řadičem

Jak již bylo zmíněno (2.4.) komunikace mezi mikroprocesorem a LCD displejem probíhá prostřednictvím řídicí a datové sběrnice. Řídicí sběrnice je pouze jednosměrná, a to směrem od mikroprocesoru k LCD modulu, ale datová sběrnice může pracovat i v opačném směru. Díky této variabilitě datové sběrnice lze nejenom zapisovat data a instrukce do LCD modulu, ale i číst data z LCD modulu a

kontrolovat jeho stav. Při změně směru toku dat na datové sběrnici je potřeba pamatovat také na změnu směrových registrů DDR(x) mikroprocesoru řady AVR.

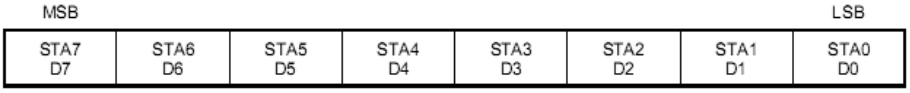
V komunikaci s LCD displeji existují dva základní principy, a to komunikace s **kontrolováním stavu řadiče** a přesné **časování komunikace**. Oba tyto principy mají své výhody a nevýhody.

2.7.1. Kontrola stavu řadiče

Princip kontroly stavu řadiče před čtením nebo zápisem dat a prováděním instrukcí spočívá v tom, že si časování celé komunikace řídí sám řadič. Využívá se specifického nastavení řídicí sběrnice (viz tabulka 3), kdy se následovně z datové sběrnice čte stavové osmibitové slovo (viz obrázek 9), kde každý bit informuje o stavu řadiče (viz tabulka 4).

Pin	C/D	WR	RD
Log. úroveň	0	1	0

tabulka 3 - nastavení řídicí sběrnice pro čtění stavu řadiče

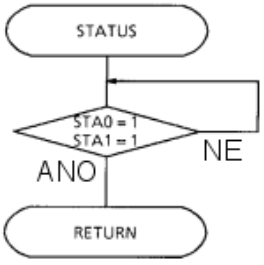


obrázek 9 - stavové slovo [3]

STA0	Vykonávání instrukcí	0: zakázáno, 1: povoleno
STA1	Čtení nebo zápis dat	0: zakázáno, 1: povoleno
STA2	Automatické čtení dat	0: zakázáno, 1: povoleno
STA3	Automatický zápis dat	0: zakázáno, 1: povoleno
STA4	Nevyužito	
STA5	Operace řadiče	0: zakázáno, 1: povoleno
STA6	Indikace chyby (pro instrukce Peek a Copy Sreen)	0: bez chyby, 1: chyba
STA7	Status displeje	0: vypnuto, 1: zapnuto

tabulka 4 - význam jednotlivých bitů stavového slova

Kontrola stavu displeje probíhá dle následujícího vývojového diagramu.



obrázek 10 - status kontrola

Výhodou tohoto způsobu komunikace je jeho nenáročnost na přesné časování a dokonalé přizpůsobení komunikace řadiči. Většina řadičů při různých provozních podmínkách (napájecí napětí, okolní teplota) nemá konstantní doby trvání provádění instrukcí. Proto je tento způsob komunikace vhodnější volit tam, kde je hlavní funkcí aplikace zobrazování dat na displeji (případ této práce).

2.7.2. Časování komunikace

Oproti principu komunikace kontrolováním stavu řadiče (2.7.1.) je způsob přesného časování komunikace poměrně složitější a vyžaduje přesné dodržení prodlev mezi zápisem dat a prováděním jednotlivých instrukcí. Jediné zjednodušení tento princip přináší ve vypuštění rutiny na kontrolování statusu řadiče (obrázek 10). Časové údaje potřebné pro sestavení přesného časování jsou většinou uvedeny katalogových listech daného řadiče.

Výhodou tohoto způsobu komunikace je nezávislost celé aplikace na LCD modulu, například při poruše LCD displeje nedojde k zastavení celé funkčnosti mikroprocesoru (čekání na status), ale pouze se přestanou zobrazovat data na LCD. Tento způsob je tedy vhodný používat například u zařízení, kde může dojit k mechanickému poškození LCD displeje, ale je nutné zachovat funkci zařízení (např. lyžařský turniket).

2.8. Inicializace LCD modulu s T6963C

Pomocí inicializace se řadič nastaví do režimu, v kterém ho budeme chtít používat. Dále je nutno v inicializaci nastavit základní hodnoty potřebné pro chod řadiče s daným LCD modulem. Inicializace většiny řadičů se dá shrnout do jednoduchého postupu:

2.8.1. Restart řadiče

Provedeme přivedením logické nuly na pin s označením RES. Výrobce doporučuje tento stav podržet po dobu pěti hodinových taktů řadiče (při 5.5MHz cca 90μs). Po přivedení logické jedničky na stejný pin se řadič aktivuje a můžeme pokračovat v dalším bodě inicializace.

2.8.2. Vymezení grafické paměti

Vymezení grafické paměti spočívá v tom, že řadiči oznámíme adresu v externí paměti RAM, od které má řadič číst data pro zobrazení na displeji. Jak již bylo napsáno, tento řadič umožňuje obsluhovat externí RAM do velikost 64kB, tudíž adresní rozsah je 0x0000 až 0xFFFF. Počátek grafické paměti můžeme libovolně zvolit v rozmezí 0x0000 až 0xF7FF, kde začíná oblast paměti určená pro uložení znaků textového režimu tzv. CG RAM. Počátek paměti je nutné volit s ohledem na rozlišení displeje. Při rozlišení 240 x 64 bodů grafická paměť jedné obrazovky obsadí 2560 bytů (viz 2.6.). Pokud LCD displej nebude využíván v grafickém ani kombinovaném režimu, není potřeba provádět toto nastavení.

2.8.3. Vymezení textové paměti

Vymezení textové paměti je podobné jako vymezení grafické paměti (2.8.2.), ale volíme jinou adresu počátku a jiný (většinou značně menší) rozsah. Pokud nebude LCD displej využíván v textovém ani kombinovaném režimu, není potřeba provádět toto nastavení.

2.8.4. Nastavení režimu zobrazení

Jak již bylo popsáno výše (2.5.) řadič umožňuje použití dvou režimů zobrazení (grafický a textový režim) a jejich kombinace. V této části nastavení oznamujeme řadiči, jaký režim budeme využívat. V této části lze také nastavit zdroj pro vygenerování znakové sady. Toto nastavení je již ale mimo rozsah této práce.

2.8.5. Nastavení zobrazení

A jako předposlední, ale velmi důležitý, bod inicializace je samostatné aktivování zobrazování na LCD displeji. Samostatně lze aktivovat či deaktivovat grafickou a textovou část zobrazovaní. Pro textovou část lze aktivovat i klasický kurzor nebo blikající kurzor. Řadič T6963C dovoluje i nastavit vzhled kurzoru.

2.8.6. Vynulování paměti

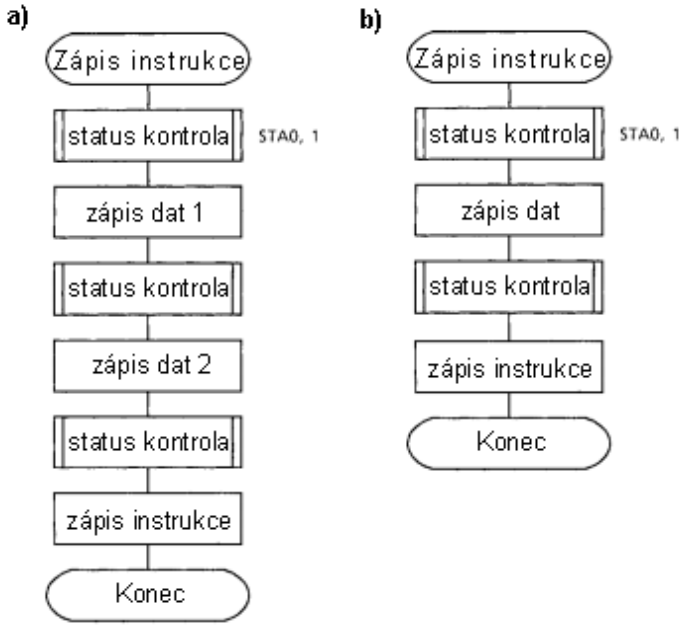
Jelikož řadič T6963C nenabízí žádnou instrukci pro mazání LCD displeje, musí se obsah paměti smazat „ručně“ zápisem hodnot 0 na všechny příslušná paměťová místa v externí RAM. Paměť se musí nulovat zvlášť pro textovou a grafickou oblast.

2.9. Instrukce

Každý řadič nabízí sadu instrukcí pomocí níž lze ovládat celý LCD displej. Pro zápis instrukce musí být nastavena řídicí sběrnice do odpovídajícího režimu (viz tabulka 5). Kód instrukce se poté zapisuje na datovou sběrnici. Nakonec je instrukce potvrzena zapsáním logické nuly na pin označený jako CE, čímž se obvod aktivuje a instrukce se začne provádět. Instrukce mohou obsahovat jeden nebo dva byty dat, která jsou při provádění instrukce zpracována. Při zápisu těchto 2 nebo 3bajtových instrukcí platí, že se první zapisují data, a to vždy první nižší významový byte následovaný vyšším významovým bytem a nakonec samotná instrukce. Mezi jednotlivými zápisy dat a instrukcí se musí kontrolovat status řadiče (2.7.1.) nebo dodržet přesně časování (2.7.2.).

pin	C/D	WR	RD
Log. úroveň	1	0	1

tabulka 5 - nastavení řídicí sběrnice pro zápis instrukce



obrázek 11 - zápis a) 3bajtové instrukce, b) 2bajtové instrukce

instrukce	kód	Data 1	Data 2	Funkce
REGISTERS SETTING	00100001 00100010 00100100	Nižší byte adresy	Vyšší byte adresy	Nastavení adresy kurzoru Nastavení textového offsetu Nastavení adresy (DAP) *
SET CONTROL WORD	01000000 01000001 01000010 01000011			Nastavení počátku text. paměti Nastavení rozsahu text. paměti Nastavení počátku graf. paměti Nastavení rozsahu graf. paměti
MODE SET	1000X000 1000X001 1000X011 1000X100 10000XXX 10001XXX			OR kombinovaný režim EXOR kombinovaný režim AND kombinovaný režim Atributy textu režim Interní zdroj CG ROM Externí CG RAM
DISPLAY MODE	10010000 1001XX10 1001XX11 100101XX 100110XX 100111XX			Displej vypnut Kurzor zapnut, blikání vypnuto Kurzor zapnut, blikání zapnuto Textový režim Grafický režim Kombinovaný režim
CURSOR PATTERN SELECT	10100000 10100001 10100010 10100011 10100100 10100101 10100110 10100111			1−řádkový kurzor 2−řádkový kurzor 3−řádkový kurzor 4−řádkový kurzor 5−řádkový kurzor 6−řádkový kurzor 7−řádkový kurzor 8−řádkový kurzor
DATA AUTO READ / WRITE	10110000 10110001 10110010			Automatický zápis dat Automatické čtení dat Automatické vypnutí
DATA READ / WRITE	11000000 11000001 11000010 11000011 11000100 11000101			Zápis dat a ADP + 1 Čtení dat a ADP + 1 Zápis dat a ADP - 1 Čtení dat a ADP - 1 Zápis dat bez změny ADP Čtení dat bez změny ADP
SCREEN PEEK	11100000			Nepoužito
SCREEN COPY	11101000			Nepoužito
BIT SET / RESET	11110XXX 11111XXX 1111X000 1111X001 1111X010 1111X011 1111X100 1111X101 1111X110 1111X111			Bod nezobrazit Bod zobrazit Bit 0 (LSB) Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 Bit 7 (MSB)

tabulka 6 - instrukce řadiče T6963C [3]

2.10. Data

Zápis i čtení dat se provádí vždy na/z adresu určenou speciálním ukazatelem (ADP – Address Data Pointer). ADP je 16bitový vnitřní registr řadiče, který lze měnit pomocí instrukce ze skupiny REGISTERS SETTINGS (viz tabulka 5). Hodnotu ADP registru lze měnit přímo při zápisu nebo čtení dat použitím odpovídajících instrukcí ze skupiny DATA READ/WRITE (viz tabulka 6).

Řadič T6963C nabízí dva režimy zápisu i čtení dat do/z externí paměti RAM a to manuální a automatický. Automatický režim je určen pro zápis celého bloku dat a umožňuje po své aktivaci zápis čistých dat na datovou sběrnici bez řídicích instrukcí (pro jednoduchost a názornost zatím nebyl v knihovně využit). V knihovně je zatím využíván pouze manuální režim, kdy se vždy musí nejprve zapsat 8bitové slovo dat a poté řídicí instrukce zápisu s požadovanou změnou ADP. Z toho vyplívá, že automatický režim má poloviční nárok na rychlost komunikace mezi řadičem a mikroprocesorem a je tedy minimálně dvakrát tak rychlejší, než manuální režim (Automatický režim je tedy více než výhodné použít například pro mazaní celého displeje).

Jelikož se jedná o grafickou knihovnu a nejmenším možným zobrazovaným prvkem je bod (pixel), s výhodou jsem využil 1 bytových instrukcí ze skupiny BIT SET / RESET, které umožňují nastavit nebo smazat pouze jediný bod (viz tabulka 6). Tyto instrukce stejně jako instrukce ze skupiny DATA READ / WRITE zapisují data na určenou adresu pomocí ADP. Proto je nutné před použitím instrukcí zápisu dat provést nastavení tohoto registru pomocí poslední instrukce ze skupiny REGISTER SETTINGS. Nová adresa pro ADP se vypočítá ze známého uspořádání externí paměti RAM (2.6.) a souřadnic kresleného bodu.

3. Software AVR LCD library

Nyní se dostáváme k samotné implementaci knihovny pro řízení grafických LCD displejů pojmenované AVR LCD library. Celá knihovna je navržena modulárně tak, aby výměna libovolného grafického LCD modulu v praxi znamenala pouze změnu názvu hlavičkového souboru ovladače LCD modulu, který obsahuje základní povinné funkce pro řízení LCD displeje.

V současné době se knihovna skládá ze tří vrstev, kde spodní vrstva představuje hlavičkový soubor daného obvodu řadiče (např. T6963C.h), která obsahuje definice dostupných instrukcí. Tuto nejnižší vrstvu využívá další vyšší vrstva, tzv. ovladač LCD modulu (např. MG2406F.h), která nabízí základní informace o LCD modulu a funkce pro ovládání celého LCD modulu, jako jsou *lcd_write*, *lcd_read*, *lcd_clear*, *lcd_init* a hlavně *pixel*. Zatím nejvyšší vrstvou ovladače AVR LCD library je hlavičkový soubor (avrlcd.h) s in-line funkcemi pro kreslení čáry, obdélníku, kružnice a jiných grafických primitivů. Jednotlivé vrstvy knihovny budou podrobně popsány v samostatných kapitolách.

3.1. Vrstva řadiče

Tuto část představuje jednoduchý hlavičkový soubor neobsahující žádnou logiku. Obsahuje pouze definice (makra) instrukcí, které nabízí k využití vyšší vrstvě (3.3.). Jelikož existuje mnoho LCD modulů, které využívají stejných obvodů řadičů, je proto vhodné mít tuto část popisující pouze funkci řadiče v samostatném souboru. Tímto krokem se nám podařilo „oddělit“ řadič od LCD modulu. Soubor vrstvy řadiče může mít následující podobu.

```
/*
 ***************************************************************************
 *                                                                         *
 *   T6963C - řadič LCD displeje                              9.11.2007  *
 *                                                                         *
 ***************************************************************************
*/
// Definice všech instrukci podporovaných tímto řadičem
//-------------------------------------------------------------
#define SET_CURSOR_POINTER                  0x21
#define SET_OFFSET_REGISTER                 0x22
#define SET_ADDRESS_POINTER                 0x24
//-------------------------------------------------------------
#define SET_TEXT_HOME_ADDRESS               0x40
#define SET_TEXT_AREA                       0x41
#define SET_GRAPHICS_HOME_ADDRESS           0x42
#define SET_GRAPHICS_AREA                   0x43
//-------------------------------------------------------------
#define MODE_OR                             0x80
#define MODE_EXOR                           0x81
#define MODE_AND                            0x83
#define TEXT_ATTRIBUTE                      0x84
#define INTERNAL_CG_ROM                     0x80
#define EXTERNAL_CG_RAM                     0x88
//-------------------------------------------------------------
#define DISPLAY_MODE                        0x90
#define SHOW_CURSOR                         0x92
#define CUSROS_BLINK                        0x91
#define DISPLAY_TEXT                        0x94
#define DISPLAY_GRAPHICS                    0x98
//-------------------------------------------------------------
#define CUSRSOR_PATERN                      0xA0
//-------------------------------------------------------------
#define AUTO_WRITE                          0xB0
#define AUTO_READ                           0xB1
#define AUTO_RESET                          0xB2
//-------------------------------------------------------------
#define WRITE_INC_ADP                       0xC0
#define READ_INC_ADP                        0xC1
#define WRITE_DEC_ADP                       0xC2
#define READ_DEC_ADP                        0xC3
#define WRITE_NO_ADP                        0xC4
#define READ_NO_ADP                         0x05
//-------------------------------------------------------------
#define SCREEN_PEEK                         0xE0
//-------------------------------------------------------------
#define SCREEN_COPY                         0xE8
//-------------------------------------------------------------
#define BIT_SET                             0xF8
#define BIT_RESET                           0xF0
```

3.2. Vrstva LCD modulu

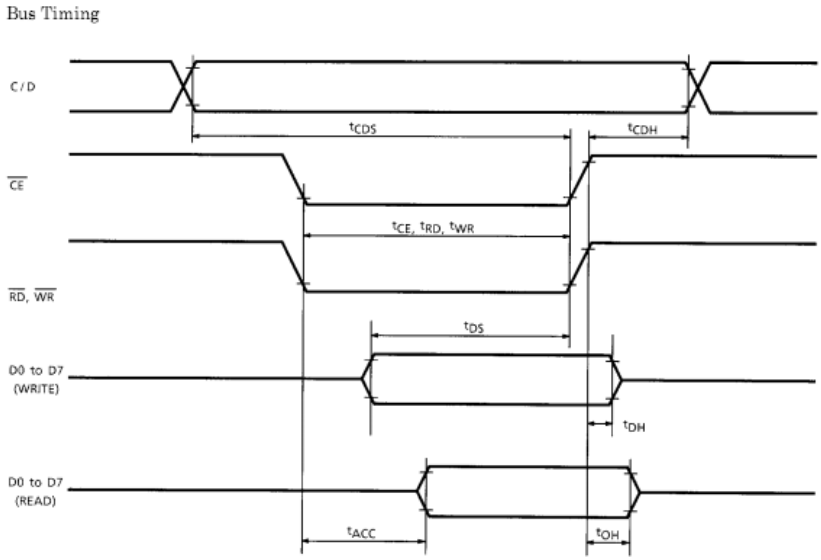
Tato vrstva knihovny již obsahuje základní funkce zprostředkovávající komunikaci s LCD řadičem a několik nadřazených funkcí pro inicializaci a vymazání displeje. Hlavní funkcí je funkce pixel, která je využívána zatím nejvyšší vrstvou *avrlcd.h* a umožňuje na libovolnou souřadnici LCD displeje vykreslit bod zadanou barvou.

3.2.1. Funkce *lcd_write*

```
void lcd_write(unsigned char lcd_ctr, unsigned char data)
```

Tato funkce má za úkol zapsat na řídicí sběrnici 8bitový parametr *lcd_ctr* a 8bitový parametr *data* na datovou sběrnici. Pomocí této instrukce se zapisují jak data(2.10.), tak i instrukce(2.9.) do řadiče. Funkce je využívána všemi ostatními

funkcemi mimo lcd_read. Zápis jednotlivých signálu na piny LCD modulu by měl probíhat v časovém uspořádání dle následujícího obrázku (viz obrázek 12).



Test Conditions (Unless Otherwise Noted, V_{DD} = 5.0 V ± 10%, V_{SS} = 0 V, Ta = −20 to 75°C)

Item	Symbol	Test Conditions	Min	Max	Unit
C / D Set-up Time	t_{CDS}	—	100	—	ns
C / D Hold Time	t_{CDH}	—	10	—	ns
$\overline{CE}$, $\overline{RD}$, $\overline{WR}$ Pulse Width	t_{CE}, t_{RD}, t_{WR}	—	80	—	ns
Data Set-up Time	t_{DS}	—	80	—	ns
Data Hold Time	t_{DH}	—	40	—	ns
Access Time	t_{ACC}	—	—	150	ns
Output Hold Time	t_{OH}	—	10	50	ns

obrázek 12 - signály na pinech LCD modulu [3]

3.2.2. Funkce *lcd_read*

```
unsigned char lcd_read(unsigned char lcd_ctr)
```

Tato funkce nastaví směr toku dat z LCD modulu do mirkoprocesoru a přečte 8bitové slovo z datové sběrnice. Této funkce se hlavně využívá při zjišťování stavu řadiče(2.7.1.), ale je možné ji také využít pro získávání obsahu externí paměti RAM. Přečtené 8bitové slovo je předáno jako návratová hodnota funkce. Jako parametr se udává nastavení řídicí sběrnice.

3.2.3. Funkce *lcd_clear*

```
void lcd_clear(unsigned char mode)
```

Funkce lcd_clear zajistí mazání (vynulování) externí paměti RAM. Jelikož řadič T6963C nedisponuje žádnou komplexní instrukcí pro mazání paměti, je tato funkce realizována jako smyčka zapisující na daný rozsah paměti hodnotu 0. Funkce využívá funkci lcd_write. Vstupní 8bitový parametr mode není zatím využit, ale v budoucnu by měl určovat režim a stránku, která bude smazána.

3.2.4. Funkce *lcd_init*

```
void lcd_init()
```

Funkce *lcd_init* inicializuje(2.8.) řadič v LCD modulu do základního pracovního režimu celého LCD modulu. Tato funkce by měla být volána vždy po zapojení displeje k napájecímu napětí a displej by měl být, po ukončení této funkce, schopen zobrazovat požadovaná data. Funkce využívá funkci *lcd_write* a nakonec provede i vymazání externí paměti RAM pomocí funkce *lcd_clear(0)*.

3.2.5. Funkce *pixel*

```
void pixel(short int x, short int y, unsigned char color)
```

Funkce pixel je nejdůležitější funkcí této vrstvy, která je dále nabízena pro využití ve vyšších vrstvách ovladače. Vstupní 16bitové parametry *x* a *y* určují souřadnice kresleného bodu na LCD displeji. Poslední 8bitový parametr *color* určuje barvu bodu (u monochromatického LCD 0 = bod nezobrazen, 1 = bod zobrazen). Funkce využívá funkce lcd_write pro nastavení ADP a zápisu instrukcí ze skupiny BIT SET / RESET (viz tabulka 6).

Kreslenou oblast lze vymezit pomocí 4 16bitových proměnných tvořící zobrazovací okno (viz tabulka 7). Zobrazovací okno by mělo být nastaveno po vykonání funkce *lcd_init (3.3.4.)* na plný rozměr LCD displeje.

Typ	Název	Funkce
`short int`	`viewport_top`	horní okraj okna
`short int`	`viewport_left`	levý okraj okna
`short int`	`viewport_right`	pravý okraj okna
`short int`	`viewport_bottom`	dolní okraj okna

tabulka 7 - definice zobrazovacího okna

Jelikož se pro funkci *pixel* používají instrukce ze skupiny BIT SET / RESET, které umožňují nastavit jednotlivý bit ze skupiny 8(6) bitů na adrese dané registrem ADP, musí se tedy před provedením této instrukce nastavit i správně ukazatel ADP. Výpočet hodnoty pro ADP probíhá dle následující ukázky:

```
unsigned short int address = (x/6) + (LCD_DOTS_X / 6) * y;
```

kde se do 16bitové proměnné *address* uloží výsledná hodnota pro registr ADP. Proměnné *x* a *y* jsou vstupní parametry funkce *pixel*, LCD_DOTS_X je definice (makro) fyzického horizontálního rozlišení LCD displeje. Velikost horizontálního rozlišení se vždy dělí 6, protože právě 6 bodů je uloženo do jedné 8bitové paměťové buňky externí paměti RAM.

Poslední zajímavou vlastností funkce pixel je použití 16bitové proměnné *last_pixel_address*, dle které se kontroluje nutnost provádění zápisu adresy do registru ADP. Toto opatření zrychluje vykreslení bodů se stejnou adresou, tedy 6 horizontálně sousedících bodů.

3.3. Grafická vrstva (canvas)

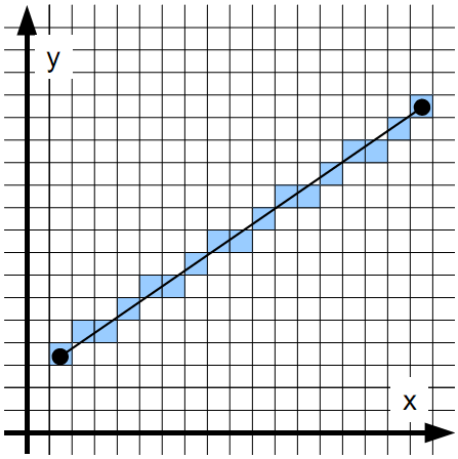
Poslední a nejvyšší vrstva ovladače vytváří již „skutečné“ kreslící plátno (canvas) pro využití programátorem. Tato vrstva využívá pouze jediné funkce z vrstvy LCD modulu (3.3.) a to *pixel* (*3.3.5.*), pomocí níž kreslí grafické útvary jakou jsou čára, trojúhelník, obdélník, polygon, kružnice a jiné obecně nazývané jako grafické primitivy. Jelikož při kreslení těchto primitivů se využívá poměrně velkého počtu bodů, je potřeba, aby výsledné algoritmy byly rychlé a nenáročné na výpočetní výkon. Proto je nutné co možná nejméně používat aritmetiku s plovoucí desetinnou čárkou, která je na platformě AVR pouze emulována a tedy značně pomalejší než aritmetika celočíselná.

3.3.1. Rasterizace úsečky

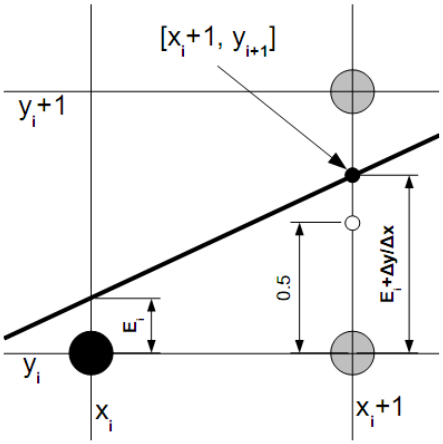
Jelikož je LCD displej tvořen konečným počtem bodů, jde tedy o tzv. rastrovou grafiku (například rastr 240 x 64 bodů). Usečka se skládá z nekonečného množství bodů mezi dvěma body, proto abychom ji mohli zobrazit na rastr (omezený počet bodů) displeje musíme provést tzv. rasterizaci. Cílem rasterizace je převést objekty vektorové grafiky co možná s nejmenší odchylkou do grafiky rastrové.

Obecná rovnice úsečky

$$Ax + By + C = 0, \text{ kde } A = (y1 - y2),\ B = (x2 - x1) \qquad (1)$$



obrázek 13 - rasterizace úsečky [4]



obrázek 14 - odchylka v zobrazení [4]

Existují různé algoritmy pro rasterizaci úsečky, ale jako nejvhodnější se jeví Bresenhamův algoritmus, protože pro výpočet využívá pouze celočíselnou aritmetiku. Princip tohoto algoritmu spočívá v tom, že se vykresluje přímka po bodu od souřadnic počátku přímky k souřadnicím konce přímky. Podle směrnice přímky se volí přírůstek ve vertikálním nebo horizontálním směru dy = 1 nebo dx = 1. O posunu v druhém směru rozhoduje tzv. znaménkový prediktor P.
Rozhodování a výpočet chyby vykreslování [6]

$$E_i + \frac{\Delta y}{\Delta x} \begin{cases} < 0.5 & krok\ (x_i + 1, y_i) \quad E_{i+1} = E_i + \frac{\Delta y}{\Delta x} \\ \geq 0.5 & krok\ (x_i + 1, y_i + 1) \quad E_{i+1} = E_i + \frac{\Delta y}{\Delta x} - 1 \end{cases} \tag{2}$$

Před porovnáním s 0.5 na test znaménka prediktoru P_i nerovnice násobíme $2\Delta x$ [4]

$$2\Delta x E_i + 2\Delta y - \Delta x \begin{cases} < 0 & E_{i+1} = E_i + 2\Delta y \\ \geq 0 & E_{i+1} = E_i + 2\Delta y - 2\Delta x \end{cases} \tag{3}$$

$$P_i = 2\Delta x E_i + 2\Delta y - \Delta x \begin{cases} < 0 & P_{i+1} = P_i + 2\Delta y \\ \geq 0 & P_{i+1} = P_i + 2\Delta y - 2\Delta x \end{cases} \tag{4}$$

Počáteční hodnota predikce pro $E_0 = 0$ pak vychází [6]

$$P_0 = 2\Delta y - \Delta x \tag{5}$$

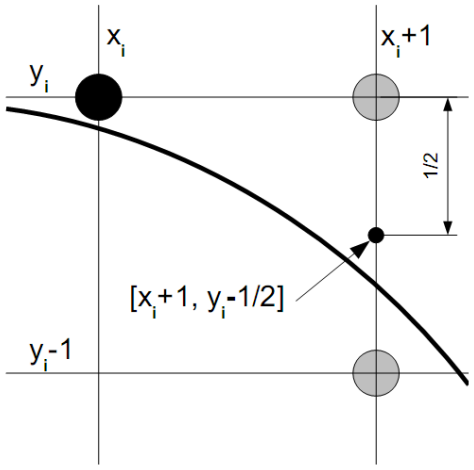
Výsledná funkce knihovny se volá v následující podobě:

```
void line(short int x1, short int y1, short int x2,
          short int y2, unsigned char color)
```

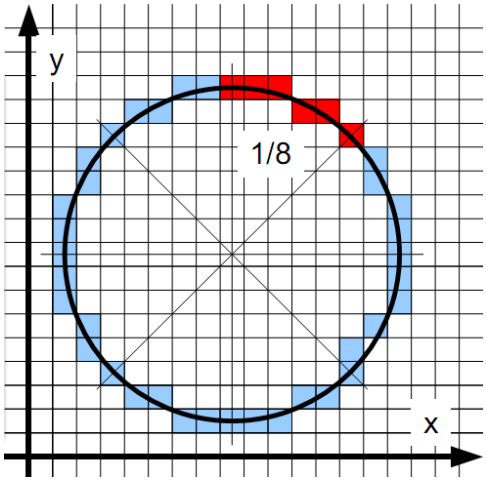
kde [*x1, y1*] je počátek úsečky, [*x2, y2*] je konec úsečky a *color* je barva úsečky. Pro kreslení úseček je také možné použít funkcí *moveto(x, y)* a *lineto(x, y, color).*

3.3.2. Rasterizace kružnice

Stejně jako úsečka, je kružnice dána nekonečným počtem bodů se stejnou vzdáleností od daného bodu (středu). Pro rasterizaci kružnice pro platformu AVR je nejvhodnější použít modifikovaný **Bresenhamův** algoritmus zvaný **MidPoint** algoritmus. Tento algoritmus také využívá pouze celočíselnou aritmetiku. Princip spočívá podobně jako u rasterizace úsečky v tom, že kreslíme kružnici po bodech od souřadnice [x, y + r] a v ose x postupujeme s přírůstkem dx = 1, posun kresleného bodu v ose y určujeme opět podle znaménka prediktoru.



obrázek 15 - rasterizace kružnice obrázek [4]



obrázek 16 - odchylka při zobrazení

Kružnice je dána souřadnicemi středu a hodnotou poloměru. Při pohledu na obrázek 15 je zřejmé, že kružnice je osově i středově souměrná. Z toho vyplývá, že pro vykreslení kružnice postačí vypočítat souřadnice pouze 1/8 bodů, z kterých se kružnice skládá. Ostatní body získáme vhodnou záměnou souřadnic a jejich znamének.

Obecná rovnice kružnice

$$(x - s_1)^2 + (y - s_2)^2 - R^2 = 0 \qquad (6)$$

odvození prediktoru u MidPoint algoritmu [4]

$$\begin{aligned} F(x.y) &= x^2 + y^2 - R^2 = 0 \qquad (7) \\ p_i &= F(x_i + 1.y_i - \frac{1}{2}) \\ p_i &= (x_i + 1)^2 + (y_i - \frac{1}{2})^2 - R^2 \\ \\ p_i &< 0 \implies y_{i+1} = y_i \\ p_i &\geq 0 \implies y_{i+1} = y_i - 1 \end{aligned}$$

funkce pro vykreslení kružnice se volá v tomto tvaru:

```
void circle(short int x, short int y, short int r,
            unsigned char color)
```

kde [*x, y*] je souřadnice středu, *r* je poloměr kružnice a *color* je barva.

3.3.3. Ostatní grafické objekty

Ostatní grafické objekty jako je trojúhelník, obdélník, polygon a jiné jsou již kresleny za použití předchozích objektů. V současné době vrstva canvas nabízí funkce pro kreslení obdélníku, vyplněného obélníku, kružnice a trojúhelníku.

Volání funkce pro kreslení trojúhelníku:

```
void triangle(short int x1, short int y1, short int x2,
              short int y2, short int x3, short int y3,
              unsigned char color)
```

kde [xn, yn] jsou souřadnice vrcholů a color barva trojúhelníku.

Volání funkce pro kreslení obdélníku:

```
void rectangle(short int x1, short int y1, short int x2,
               short int y2, unsigned char color)
```

kde [*x1, y1*] je souřadnice levého horního rohu, [*x2, y2*] je souřadnice pravého dolního rohu obdélníku a parametr *color* určuje barvu.

Poznámka: tato funkce umožňuje pomocí parametru color = 2 kreslit výsledný objekt tečkovaně.

Volání funkce pro kreslení vyplněného obdélníku:

```
void fillrect(short int x1, short int y1, short int x2,
              short int y2, unsigned char pen,
              unsigned char brush)
```

kde [*x1, y1*] je souřadnice levého horního rohu, [*x2, y2*] je souřadnice pravého dolního rohu obdélníku, parametr *pen* určuje barvu obrysu obdélníku a parametr *brush* určuje barvu.

3.4. Rozšiřující moduly

Jak již bylo výše zmíněno, byla celá knihovna AVRLCD navržena jako modulární a umožňuje tak svůj vlastní rozvoj pomocí dalších uživatelských modulů. Cílem této práce bylo implementovat pro mikroprocesory řady AVR sadu základních ovládacích prvků jako jsou například tlačítka, checkboxy a jiné prvky známé například z operačního systému Windows.

3.4.1. Grafické texty – Bitmap Font

Jelikož použitý řadič grafických displejů od firmy Toshiba s označením T6963c má integrovaný generátor znakové sady pouze pro textový režim, bylo nezbytně nutné před začátkem tvoření samostatných ovládacích prvků vytvořit vlastní generátor znaků a textů pro grafický režim, a to nejlépe nezávislý na typu použitého řadiče. V podstatě se nabízely dvě možnosti jak tento problém vyřešit.

Kopírování znaků z textové paměti do grafické

Tento způsob spočívá v tom, že se využije integrovaný generátor znakové sady a požadovaný znak se vyčte z CG RAM(viz 2.5.1.) LCD modulu. Takto získaná data se dále zpracují odpovídající rutinou a nakonec se zapíší na požadované místo v paměti přidělené pro grafický režim.

výhody
- \+ malá náročnost na paměť mikroprocesoru (znaková sada je uložena v LCD modulu)
- \+ není potřeba vytvářet novou znakovou sadu

nevýhody
- – rychlost kreslení znaků (pomalé vyčítání znaků z LCD modulu)
- – pevně daná znaková sada
- – závislost na řadiči displeje (ne každý řadič má generátor znakové sady)

Použití vlastní znakové sady (Bitmap Font)

Způsob použití vlastní znakové sady spočívá ve vytvoření vlastní prezentace jednotlivých znaků, které jsou uloženy v programové paměti FLASH mikroprocesoru. Takto uložená data jsou pak zpracována odpovídající rutinou a kreslena bod po bodu na displej s použití funkce pixel (viz 3.2.5.).

Pro tento účel byl specifikován nový formát dat znakové sady, který umožňuje použití libovolné velikosti fontu (maximálně o rozměrech 256x256 bodů).

výhody
- \+ rychlost kreslení znaků (pouze jednosměrná komunikace s LCD modulem)
- \+ nezávislost na typu řadiče LCD displeje
- \+ uživatelsky definovatelná znaková sada
- \+ definovatelná velikost znaků (až 256x256 bodů)
- \+ možnost použití více znakových sad současně
- \+ proměnná šířka znaku (šetření místa na displeji)

nevýhody
- – větší náročnost na paměť mikroprocesoru

Z výše uvedeného soupisu výhod a nevýhod obou metod kreslení znaků a textů na grafický displej je zřejmé, že metoda s použitím vlastní znakové sady je výhodnější. Proto jsem se rozhodl použít tuto metodu a vytvořil rozšiřující modul pro kreslení grafických textů nazvaný *avrlcd_fonts.h.* Princip a funkce tohoto modulu je popsána v následující kapitole (3.4.2.).

3.4.2. Rozšiřující knihovna AVRLCD_FONTS

Rozšiřující knihovna AVRLCD_FONTS nabízí 3 základní funkce pro vykreslení textu na grafický LCD displej *draw_char*, *draw_text* a *draw_text_ex* a 4 pomocné funkce *get_char_height*, *get_char_width*, *get_text_width* a *get_text_height*. Pomocí těchto sedmi funkcí je možné zjistit potřebné informace o kresleném textu a vykreslit požadovaný text na displej.

3.4.2.1. Datový formát znakové sady

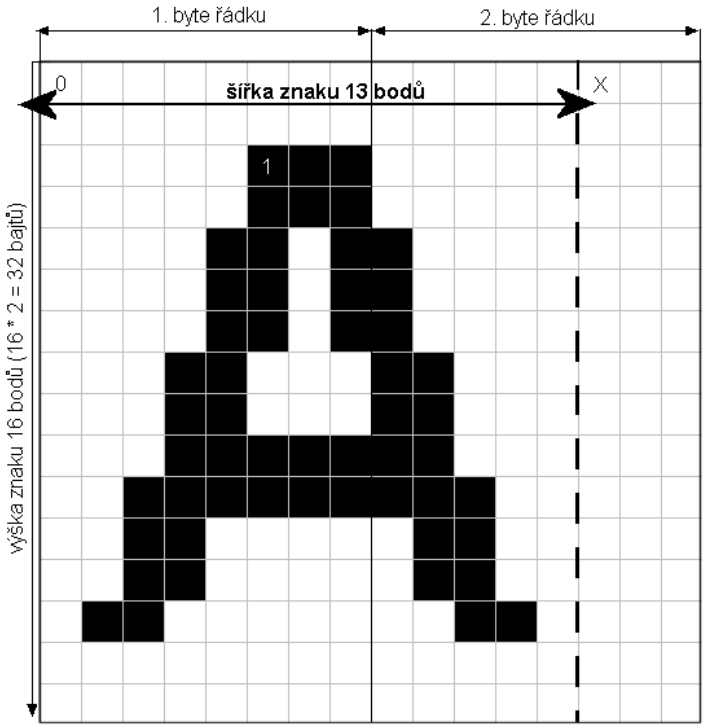
Celá znaková sada je uložena do programové paměti FLASH mikroprocesoru v přesně definovaném formátu, který je potřeba pro správnou funkčnost modulu dodržet. Data znakové sady jsou rozděleny na dvě části a to hlavičku a samotná data.

Hlavička obsahuje vždy 225 bajtů, kde první byte udává výšku fontu (maximálně 256 bodů) a zbývajících 224 bajtů pak prezentuje šířku jednotlivých znaků počínaje znakem mezery s ASCI kódem 32. Znaková sada je kompatibilní se standardní ASCI znakovou sadou známou z PC.

Datová část začíná vždy na 226 bajtu, kde je binárně prezentován obraz prvního znaku znakové sady standardně znak mezery ASCI 32. Jelikož tento formát dat nemá pevně stanovenou velikost znaku, je velikost datové části proměnlivá v závislosti na velikosti každého znaku. Jednotlivé znaky jsou ukládány jako bitová mapa po řádcích a tedy platí, že pokud je šíře znaků větší než 8 bodů, je automaticky použit další byte pro ten samý řádek a tak dále až do šíře 256 bodů, kdy je potřeba 32 bajtů na jeden řádek znaku.

Jestliže víme, že některá znaky nebudou nikdy použity v naší aplikaci je možné je vyřadit ze znakové sady a tak ušetřit paměť mikroprocesoru. Znak, který není obsažen ve znakové sadě, je prezentován jako znak s nulovou šířkou v hlavičce

datového formátu. Na následujícím obrázku je znázorněna bitová maska znaku „A“ o rozměrech 13 x 16 bodů.



obrázek 17 - bitová maska znaku "A"

Pro usnadnění práce při vytváření potřebné datové struktury znakové sady jsem vytvořil jednoduchý program FontGen, který je přiložen na CD k této práci. Tento program umožňuje přímo převést True Type fonty z Windows do odpovídajícího formátu znakové sady pro AVRLCD_FONTS. Výsledek programu je obsah hlavičkového souboru pro jazyk C v následujícím formátu.

```
/*
  AVRLCD FontGen ©2008 by Miroslav Skopal
  FontName: Small Fonts
  Height: 11
  Created: 25.5.2008 20:55:41
*/

const unsigned char PROGMEM font[] = {
  // Font Height 1.byte určující výšku všech znaků znakové sady
  0x0B,
  // Font char widths 224 bajtů, které určují šířky znaků ve  znakové sadě
  0x02/* */, 0x02/*!*/, 0x04/*"*/ … ,0x02/*˙*/,
  // Width: 2px, ASCI:  , DEC(32), HEX(0x20)
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  // Width: 2px, ASCI: !, DEC(33), HEX(0x21)
  0x00, 0x00, 0x40, 0x40, 0x40, 0x40, 0x40, 0x00, 0x40, 0x00, 0x00,
  .
  .
  . datová oblast obsahující bitové masky všech znaků
  .
  .
};
// Font FLASH size: 2722 Bytes
```

3.4.2.2. Funkce *draw_char*

Základní funkce modulu AVRLCD_FONTS, které umožňuje vykreslení jednoho znaku znakové sady. Tato funkce využívá pomocných funkcí *get_char_widht*, *get_char_height* a systémové knihovny *pgmspace.h* pro čtení dat z programové paměti mikroprocesoru. Zápis funkce je v následujícím tvaru

```
unsigned char draw_char(short int x, short int y,
                        unsigned char ch, unsigned char tags)
```

, kde 16 bitové proměnné *x* a *y* určujíc souřadnice kresleného znaku a 8 bitová proměnná *ch* specifikuje znak pro vykreslení. Poslední 8 bitový parametr funkce tags v tomto případě slouží jako přepínač reagující na parametry *DT_INVERT* nebo *DT_NORMAL*, kdy při parametru *DT_NORMAL* je text kreslen klasicky černou barvou a při parametru *DT_INVERT* inverzně.

Funkce vrací 8 bitovou hodnotu o velikosti šířky kresleného znaku.

3.4.2.3. Funkce *draw_text_ex*

Funkce draw_text_ex umožňuje vykreslení libovolného textu do definované oblasti na displeji s možností vertikálního a horizontálního posunu v rámci této oblasti. Funkce využívá předešlé dokumentované funkce *draw_char* a reaguje na řídící znaky konce řádku \n a \t pro tabulátor.

```
void draw_text_ex(short int x, short int y, short int width,
                  short int height, short int offset_x,
                  short int offset_y, char* text,
                  unsigned char tags)
```

První čtveřice 16 bitových parametrů *x, y, width* a *height* vymezují obdélníkovou oblast na displeji kam se zadaný text vykreslí. Další dva 16 bitové parametry *offset_x* a *offset_y* určují vertikální a horizontální posun textu uvnitř této obdélníkové oblasti. Parametr *text* je ukazatel na kreslený text zakončený znakem ASCI 0 (null-terminated).

Poslední 8 bitový parametr *tags* upřesňuje použití této funkce s následujícími parametry:

- DT_NORMAL – základní nastavení černý text bílé pozadí
- DT_INVERT – inverzní zobrazení bílý text černé pozadí
- DT_TRANSPARENT – průhledné pozadí (kreslí se pouze body textu)
- DT_NOCLIP – text není ořezán při přesahu vymezené oblasti
- DT_FOCUSED – kolem kreslené oblasti nakreslí tečkovaný obdélník

3.4.2.4. Funkce *draw_text*

Tato funkce slouží pouze jako zjednodušený odkaz na předchozí funkci draw_text_ex, která jsou předány všechny zadané parametry a parametr offset_x a offset_y jsou nulové.

```
void draw_text (short int x, short int y, short int width,
                short int height, short int offset_x,
                unsigned char tags)
```

Popis parametrů viz kapitola 3.4.2.3..

3.4.2.5. Funkce *get_char_width*

Tato pomocná funkce slouží hlavně pro interní použití funkce draw_char, ale samozřejmě ji lze také využít v samotné uživatelské aplikaci. Jako jediný vstupní parametr této funkce je 8 bitová proměnná ch, která specifikuje znak, pro který chceme zjistit jeho šířku v grafických bodech. Tato hodnota je pak přečtena z hlavičky datové struktury znakové sady a vrácena jako 8 bitový výstup funkce.

```
unsigned char get_char_width(unsigned char ch)
```

Poznámka: Pokud byl zadán znak, který není obsažen ve znakové sadě návratová hodnota bude nulová.

3.4.2.6. Funkce *get_char_height*

Stejně jako funkce get_char_width slouží tato funkce pro interní použití ostatních funkcí tohoto modulu AVRLCD_FONTS. Jediný rozdíl mezi touto funkcí a funkcí get_char_width je, že tato funkce nevyžaduje žádný vstupní parametr, protože všechny znaky znakové sady musí být stejně vysoké. V praxi tedy tato funkce vždy přečte a vrátí první byte z hlavičky aktuální znakové sady.

```
unsigned char get_char_height()
```

3.4.2.7. Funkce *get_text_width*

Jak již bylo zmíněno v úvodu kapitoly a modulu AVRLCD_FONTS (3.4.2.), tak tento modul pracuje se znakovou sadu s proměnlivou šířkou znaků. Z toho důvodu bylo potřeba vytvořit pomocnou funkci, která by dokázala zjistit šířku kresleného textu. Pro tento účel byla naprogramována funkce *get_text_width*, která má jako jediný vstupní parametr ukazatel na text, u kterého chceme zjistit jeho celkovou šířku. Výstupem této funkce je pak 16 bitová hodnota určující celkovou šíři textu v grafických bodech.

```
unsigned short int get_text_width(char *text)
```

3.4.2.8. Funkce *get_text_height*

Funkce *get_text_height* je vstupními a výstupními parametry totožná z funkcí *get_text_width*. Tato funkce pouze spočítá výskyt znaku konce řádku \n v zadaném textu a vynásobí tento počet+1 odpovídající výškou znakové sady.

```
unsigned short int get_text_width(char *text)
```

3.4.2.9. Použití modulu AVRLCD_FONTS

Pro správnou funkci tohoto modulu je potřeba splnit tři základní kroky a to vytvořit znakovou sadu v odpovídajícím formátu dat (3.4.2.1.), přilinkovat ji do výsledného projektu , přilinkovat knihovnu *avrlcd_fonts.h* a přiřadit do proměnné current_font odkaz na aktuální font.

Výsledný projekt pak může vypadat následovně:

```
/*
                              AVRLCD_FONTS_DEMO
                      Program:    6314 bytes (38.5% Full)
                      Data:        114 bytes (11.1% Full)
*/
#include "avrlcd_conf.h"  // konfigurace AVR LCD knihovny
#include "MG2406F.h"      // Použity LCD modul
#include "avrlcd_fonts.h" // knihovna pro práci s bitmap fonty
#include "font.h"         // datová struktura fontu

int main(){
  lcd_init();             // inicializace displeje

  current_font = font13;  // přiřazení aktuálního fontu

  draw_text(10,10, LCD_DOTS_X - 20, 12, "Test vykreslení BitMap Fontu",
    DT_NORMAL);

  draw_text(10,25, LCD_DOTS_X - 20, 12,
    "Test vykreslení BitMap Fontu DT_INVERT", DT_INVERT);

  draw_text(10,40, LCD_DOTS_X - 20, 12, "Čeština: +ěščřžýáíé+ĚŠČŘŽÝÁÍÉ",
    DT_NORMAL);

  while(1);

}
```

obrázek 18 - výsledek ukázkové aplikace Bitmap Font

3.4.3. Rozšiřující knihovna AVRLCD_CTRLS

Tato knihovna je v rámci mé bakalářské práce doposud nejvyšší nádstavbou knihovny AVRLCD a pro svůj chod vyžaduje všechny zmíněné současti. Knihovna jako takové obsahuje grafické prezentace 6 základních ovládacích prvků a to tlačítko (button), textové pole (edit), zatrhávací pole (checkbox a radiobutton), rolovací seznam (listbox) a ovládací prvek nabídky (menu).

Při tvorbě tohoto doplňkového modulu se vyskytly 2 směry, kterými se celá knihovna mohla ubírat. Hlavním rozdílem mezi těmito odlišnými přístupy bylo samotné ovládání kontrolních prvků uživatelem.

První zamýšlený postup uvažoval plné řízení všech prvků samotnou knihovnou AVRLCD_CTRLS. Avšak tento přístup by vyžadoval nemalé paměťové nároky na programovou paměť mikroprocesoru, ale hlavně by znemožnil úplné využití všech možností co tato knihovna přináší. Poslední nevýhodou tohoto přístupu by byla složitá interakce mezi modulem a fyzicky realizovaným ovládacím rozhraním, která bývá často řešeno odlišně podle potřeby zadání.

Druhý postup spočívá v tom, že tato knihovna nabídne pouze grafické znázornění ovládacích prvků a jejich stavů, a výsledné chování již bude řízeno uživatelskou aplikací. Tento princip samozřejmě přináší určité ztížení práce uživatele, ale zároveň je méně náročný na programovou paměť mikroprocesoru a nechává otevřenou cestu k využití všech možností tohoto modulu.

3.4.3.1. Tlačítko (Button)

Tlačítko je základní ovládací prvek bez kterého se neobejde žádné grafické rozhraní. V této knihovně je ovládací prvek tlačítka prezentován jako jednoduchý obdélník s definovatelnou šířkou a výškou. Do tohoto obdélníku je vykreslen text, pomocí modulu AVRLCD_FONTS (3.4.2.), který je vertikálně i horizontálně zarovnán na střed. Tento ovládací prvek podporuje tři stavy a to základní (CTRL_NORMAL), aktivní (CTRL_FOCUSED) a stav kdy je tlačítko zmáčknuté (CTRL_PRESSED). Tyto stavy se mohou navzájem kombinovat pomocí logického operátoru OR a předávají se do funkce pomocí 8 bitového parametru *tags*.

```
void ctrl_button(short int x, short int y, short int width,
                 short int height, char* caption,
                 unsigned char tags)
```

Význam jednotlivých parametrů je vysvětlen na následujícím obrázku.

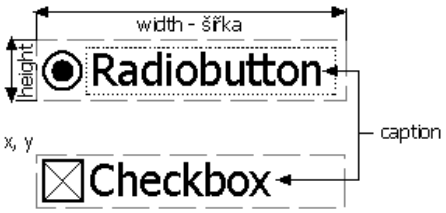
obrázek 19 - prvek tlačítka

Tečkovaný obdélník uvnitř tlačítka indikuje, že je tento prvek vybrán jako aktivní. Tohoto efektu lze dosáhnout předáním hodnoty CTRL_FOCUSED do parametru *tags*. Stín tlačítka je zase ovlivňován hodnotou CTRL_PRESSED, která indikuje že se mát tlačítko vykreslit jako stisknuté, tedy bez stínu a posunuté o jeden bod doprava a dolů. Parametr *caption* je ukazatel na text, který bude zobrazen uprostřed tlačítka.

3.4.3.2. Zatrhávací pole (Checkbox a Radiobutton)

Zatrhávací pole checkbox a radiobutton si jsou vlastnostmi i vzhledem velice blízké. Rozdíl mezi těmito ovládacími prvky spočívá v jejich chování, kdy prvek checkbox je používán jako zatržení možnosti ano či ne naopak od prvku radiobutton, který se převážně používá na výběr jedné z více možností prezentovaných několika prvky pod sebou. Jak již bylo popsáno výše chování a ovládání všech prvků bylo přenecháno na samotném programátorovi a tak lze tento odstavec brát pouze jako doporučení.

Podobně jako ovládací prvek tlačítka i tento prvek má 3 stavy a to základní (CTRL_NORMAL), aktivní (CTRL_FOCUSED) a stav kdy je prvek zatržen (CTRL_CHECKED).



obrázek 20 - Radiobutton a Checkbox

```
void ctrl_radio(short int x, short int y, short int width,
                short int height, char *caption,
                unsigned char tags)

void ctrl_checkbox(short int x, short int y, short int width,
                   short int height, char *caption,
                   unsigned char tags)
```

3.4.3.3. Textové pole (Edit)

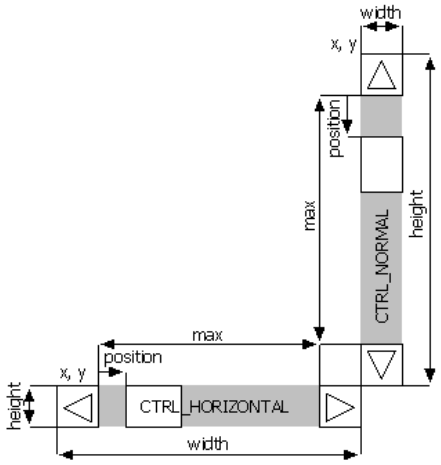
Textové pole slouží pro průběžné zobrazování zadávaného textu od uživatele například prostřednictvím klávesnice. Tento ovládací prvek nemá žádná jiný stav než základní (CTRL_NORMAL), ale z důvodu dalšího možného rozšíření byl parametr *tags* u této funkce ponechán. Čtveřice 16 bitových parametrů *x, y, width a height* mají stejnou funkci jako u ostatních ovládacích prvků. Parametr *text* je opět ukazatel na data, která mají být zobrazena.

obrázek 21 - textové pole

```
void ctrl_edit(short int x, short int y, short int width,
               short int height, char *text,
               unsigned char tags)
```

3.4.3.4. Rolovací lišta (Scrollbar)

Tento ovládací prvek se běžně samostatně moc nevyužívá, většinou slouží jako indikátor posunu textu u dalších složitějších ovládacích prvků. I v tomto modulu byl ovládací prvek scrollbar vytvořen hlavně pro využití v následující komponentě Listbox. Jelikož si uživatel může vytvořit vlastní ovládací prvky je více než vhodné mu umožnit použití i tohoto pomocného ovládacího prvku. Rolovací lišta je definované jako obdélník s dvěma tlačítky na každém konci a posuvníkem mezi nimi. Parametr *tags* zde určuje jestli se jedná o vertikální (CTRL_NORMAL) nebo horizontální (CTRL_HORIZONTAL) rolovací lištu.



obrázek 22 - rolovací lišty

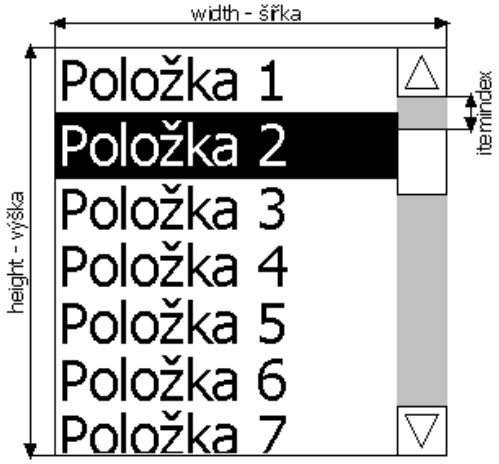
```
void ctrl_scrollbar(short int x, short int y, short int width,
                    short int height, short int max,
                    short int position, unsigned char tags)
```

Parametry x, y, width a height mají stejnou funkci jako u ostatních ovládacích prvků, zatímco dva nové parametry *position* a *max* určují samotné nastavení rolovací lišty. Jako parametr max se předáva 16 bitová hodnota, která udává maximální polohu posuvníku u vertikálního shora dolů a u horizontálního zleva doprava. Parametr position pak určuje aktuálního polohu pro vykreslení posuvníku. Poloha je vypočtena relativně k maximální hodnotě a skutečné šířce nebo výšce rolovací lišty.

3.4.3.4. Seznam (Listbox)

Komponenta seznamu (Listbox) slouží převážně k zobrazení více informací v řádcích pod sebou nebo k výběru jednoho řádku ze seznamu. Jednotlivé řádky se zobrazují pod sebou v oblasti vymezené rozměry ovládacího prvku parametry *width* a *height*. V případě, že počet řádku vynásobený výškou použitého fontu přesahuje zvolenou výšku ovládacího prvku (*height*) je na pravé straně zobrazena vertikální rolovací lišta indikující aktuální pozici v seznamu. Tato pozice se určuje 16 bitovým parametrem itemindex. Pokud si nepřejete označit jakýkoliv řádek lze použít pro tento parametr hodnotu -1.

Pro správné zobrazení jednotlivých položek (řádků) seznamu je nutné dodržet formát zadaných dat pomocí parametru *items*. Parametr *items* je opět ukazatel na text, který udává jednotlivé řádky oddělené znakem nového řádku \n.



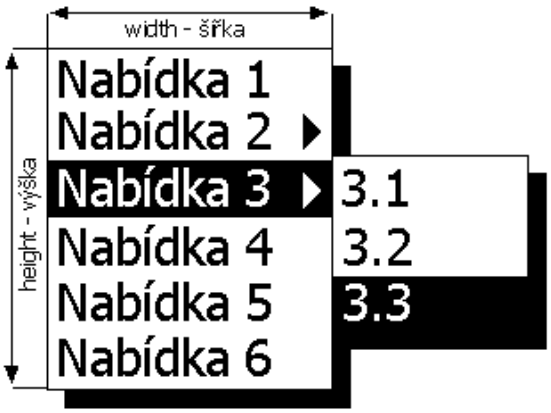
obrázek 23 - Seznam (Listbox)

```
void ctrl_listbox(short int x, short int y, short int width,
                  short int height, char* items,
                  short int itemindex, unsigned char tags)
```

Parametr *tags* není u této funkce využit.

3.4.3.5. Nabídka (Menu)

Komponenta nabídky (Menu) je z důvodů šetření zdrojů odvozena od předchozího prvku seznamu (Listbox). Prvek menu má totožné parametry i stejný styl použití jako komponenta listbox. Jediné tři rozdíly ve vhledu a chování těchto dvou ovládacích prvků je, absence rolovací lišty a přítomnost stínu u prvku nabídky a možnost přidání zobrazení šipky pro otevření vnořené nabídky. Tato šipka se zobrazuje vždy napravo v odpovídajícím řádku. Označení přítomnosti vnořené nabídky (šipky) musí uživatel vyvolat sám, pomocí vložení znaku „>" nakonec odpovídajícího řádku.



obrázek 24 - Nabídka (menu)

```
void ctrl_menu(short int x, short int y, short int width,
               char* items, short int itemindex,
               unsigned char tags)
```

Parametr *tags* není u této funkce využit.

3.4.3.6. Použití modulu AVRLCD_CTRLS

Jelikož je nutné v ovládacích prvcích zobrazovat grafický text je nezbytně nutné použití předchozího modulu AVRLCD_FONTS, na kterém je tento modul přímo závislý. Použití modulu AVRLCD_FONTS najdete v kapitole 3.4.2.9.. Jakmile splníte všechny požadavky pro použití modulu kreslení grafických textů, stačí do vašeho projektu přilinkovat knihovnu avrlcd_ctrls.h a můžete začít využívat grafické prezentace výše popsaných ovládacích prvků.

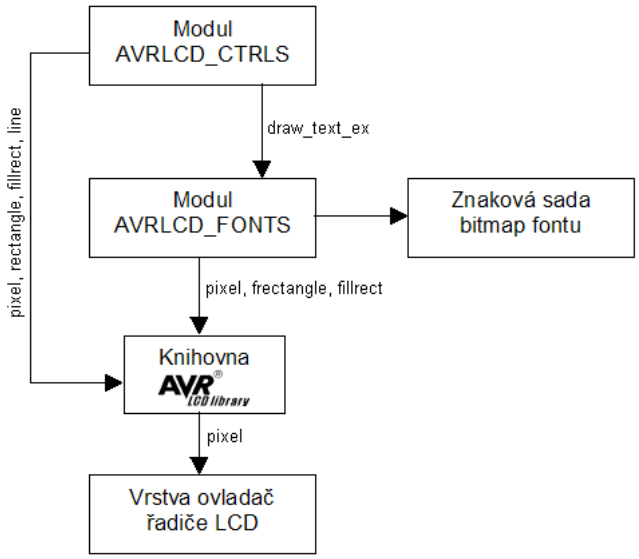
Výsledný projekt pak může vypadat následovně:

```
/*
      AVRLCD_CTRLS_DEMO
      Program:   13496 bytes (82.4% Full)
      Data:        174 bytes (17.0% Full)
*/
#include "avrlcd_conf.h"      // konfigurace AVR LCD knihovny
#include "MG2406F.h"          // Pouzity LCD

#include "avrlcd_ctrls.h"     // knihovna pro praci s ovladacimi prvky
#include "font.h"             // datova struktura

int main(){
  lcd_init();                 // incializace displeje
  current_font = font13;      // prirazeni aktualniho fontu
  ctrl_button(5, 5, 50, 15, "Button 1", CTRL_NORMAL);
  ctrl_button(5, 25, 50, 15, "Button 2", CTRL_FOCUSED);
  ctrl_button(5, 45, 50, 15, "Button 2", CTRL_FOCUSED | CTRL_PRESSED);
  ctrl_edit(60, 5, 55, 12, "Edit", CTRL_NORMAL);
  ctrl_checkbox(60, 25, 50, 15, "Checkbox", CTRL_FOCUSED | CTRL_CHECKED);
  ctrl_radio(60, 45, 50, 15, "Radiobutton", CTRL_FOCUSED | CTRL_CHECKED);
  ctrl_listbox(120, 5, 55, 55,
    "Položka 1\nPoložka 2\nPoložka 3\nPoložka 4\nPoložka 5\n", 1,
    CTRL_NORMAL);
  ctrl_menu(180, 5, 55,
    "Nabídka 1\nNabídka 2>\nNabídka 3\nNabídka 4>\nNabídka 5\n", 3,
    CTRL_NORMAL);
  while(1);
}
```

obrázek 25 - ukázka ovládacích prvků

3.5. Závislosti mezi moduly

V předchozím textu bylo několikrát naznačeno, že mezi jednotlivými moduly existují určité závislosti. Mnou vytvořené moduly AVRLCD_FONTS a AVRLCD_CTRLS jsou schopny si pomocí maker tyto závislosti ohlídat a případně přilinkovat potřebné hlavičkové soubory s in-line funkcemi. Na následující obrázku 26 jsou znázorněny závislosti těchto dvou modulů mezi sebou a na základní knihovně AVRLCD.



obrázek 26 - závislosti mezi moduly

3.6. Použití AVR LCD library v programu

Knihovna je pro jednoduchost napsána včetně funkcí pouze v hlavičkových souborech (*.h). Pro použití tedy stačí přilinkovat odpovídající soubory knihovny do vlastního souboru projektu (*.c) a zavolat funkcí *lcd_init*. Pozor na dodržení pořadí linkování jednotlivých souborů knihovny!

```
#include "avrlcd_conf.h"      // konfigurace AVR LCD knihovny
#include "MG2406F.h"          // Použitý LCD modul
#include "avrlcd.h"
```

Soubory *MG2406F.h* a *avrlcd.h* a jejich funkce jsou popsány v kapitolách 3.3. a 3.4. První linkovaný soubor *avrlcd_conf.h* obsahuje základní konfiguraci celé knihovny.

Jeho obsah vypadá následovně:

```
/*
        avrlcd_conf.h - konfigurace knihovny
*/
#include <avr/io.h>

#define LCD_DB_BUS_DDR DDRA // rizeni datoveho smeru
#define LCD_DB_BUS_OUT PORTA //datova sbernice pro zapis
#define LCD_DB_BUS_IN PINA // datova sbernice pro cteni
#define LCD_CTR_DDR DDRB // rizeni datoveho smeru
#define LCD_CTR_BUS PORTB // ridici sbernice
#define LCD_CTR_RST _BV(PIN1) // pin pro RESET
#define LCD_CTR_CD _BV(PIN2) // pin pro C/D (COMMAND/DATA)
#define LCD_CTR_RD _BV(PIN3) // pin pro READ
#define LCD_CTR_WR _BV(PIN4) // pin pro WRITE
#define LCD_CTR_CE _BV(PIN5) // pin pro CHIP ENABLED
```

Toto nastavení nemusí být v samostatném souboru, ale stačí, pokud se definují příslušné signálové piny před linkováním souboru ovladače LCD modulu (např. *MG2406F.h*).

4. Závěr

Během semestrální práce se mi podařilo vytvořit základ pro knihovnu nazvanou „AVR LCD library“, pro řízení grafických LCD displejů s možností použití různých obvodů řadičů LCD displejů. Při realizaci bylo využito již několik málo algoritmů pro zrychlení a zefektivnění práce s grafickými displeji jako například **Bresenhamův** algoritmus pro rasterizaci úsečky, **MidPoint** algoritmus pro rasterizaci kružnice nebo optimalizovaný algoritmus pro zápis sousedních bodů v řádku (3.3.5.).

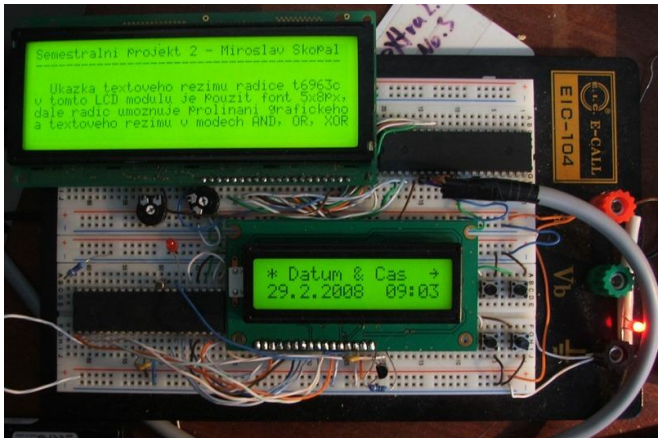
Dále bylo při tvorbě ukázkových programu použito zobrazení bitmapy uložené v programové FLASH paměti, kde bylo nutné, kvůli nedostatku operační paměti RAM, použít speciálních funkcí z knihovny *pgmspace.h* pro přímé čtení z FLASH paměti.

Celý semestrální projekt 2 byl především zaměřen na zvládnutí základních operací s grafickým LCD displejem, seznámení s algoritmy pro rasterizaci a ověření funkčnosti návrhu vícevrstvé knihovny AVR LCD library.

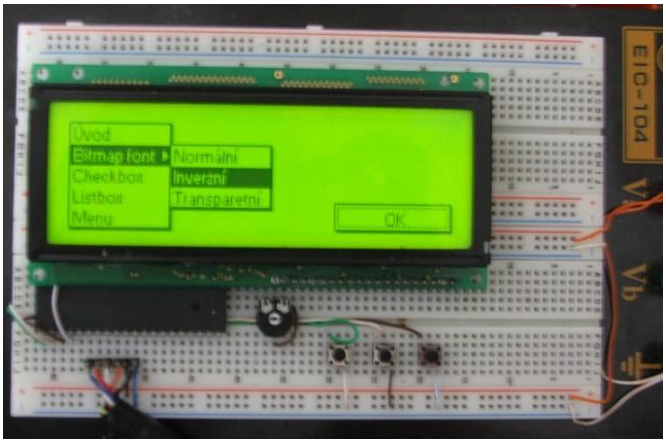
V rámci předmětu bakalářská práce byly navrženy a realizovány dva rozšiřující moduly AVRLCD_FONTS pro vykreslování bitmapových fontů a AVRLCD_CTRLS jako soubor grafických prezentací základních ovládací prvků jako je tlačítko (button), seznam(listbox) a další. Moduly nabízejí pouze základní funkce nezbytně nutné k realizování jednoduchého grafického uživatelského rozhraní (GUI-Graphics user interface), a to z důvodu nedostatku programové paměti u mikroprocesoru ATMega16, která je pouze 16 kilobajtů. Z jednotlivých demonstrací použití knihovny v kapitolách 3.4.2.9. a 3.4.3.6. je patrné, že potřeba programové paměti se pohybuje u aplikace s použitím modulu AVRLCD_FONTS přes 6 kilobajtu a s použitím modulu AVRLCD_CTRLS již přes 13 kilobajtů. Z tohoto důvodu by bylo vhodnější použití vyššího modelu mikroprocesoru s větší programovou pamětí, například ATMega32 nebo ATMega64.

Realizace další vrstvy ovladače grafických displejů od formy EPSON, jež byla naplánována v rámci předmětu semestrálního projektu 2, nebyla z důvodu mechanického poškození LCD modulu dokončena.

Dosavadní funkčnost knihovny byla testována na mikroprocesoru ATMega16 a grafickém LCD displeji MG2406F v zapojení na nepájivém kontaktním poli (viz obrázek 27 a 28). Zdrojové kódy celé knihovny a tři demonstrační aplikace pro AVR Studio jsou přiloženy k této práci na CD nosiči.



obrázek 27 - vývojová deska



obrázek 28 - demonstrační přípravek

5. Použitá literatura

[1] MATOUŠEK, D. Práce s mikrokontroléry Atmel AVR. Praha: BEN - technická literatura, 2003.

[2] MANN, B. C pro mikrokontroléry. Praha: BEN - technická literatura, 2003.

[3] *T6963C* [online]. 1998 [cit. 2008-06-03]. Dostupný z WWW: <http://www.datasheetcatalog.com/datasheets_pdf/T/6/9/6/T6963C.shtml>

[4] KRŠEK, Přemysl, ŠPANĚL, Michal. Základy počítačové grafiky : Rasterizace objektů ve 2D. *Základy počítačové grafiky* [online]. 2007 [cit. 2008-06-02].

[5] Rasterizace. *Wikipedia : Internetová encyklopedie* [online]. 2008 [cit. 2008-06-03]. Dostupný z WWW: <http://cs.wikibooks.org/wiki/Rasterizace>.

[6] Úsečka. *Wikipedia : Internetová encyklopedie* [online]. 2008 [cit. 2008-06-03]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/%C3%9Ase%C4%8Dka>.

[7] ATMega16 datasheet. *Datasheets* [online]. 2008 [cit. 2008-06-03]. Dostupný z WWW: <http://www.atmel.com/dyn/resources/prod_documents/doc2466.pdf>.

[8] Displej z tekutých krystalů. *Wikipedia : Internetová encyklopedie* [online]. 2008 [cit. 2008-06-03]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Displej_z_tekut%C3%BDch_krystal%C5%AF>.

[9] MG2406F. *Datasheets* [online]. 2008 [cit. 2008-06-02]. Dostupný z WWW: <http://www.everbouquet.com.tw/MG2406F.htm>.