TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Implementace knihovny funkcí pro grafický displej

Diplomová práce

Studijní program: N2301 – Strojní inženýrství
Studijní obor: 2301T049 – Výrobní systémy a procesy

Autor práce: **Bc. Jakub Bláha**
Vedoucí práce: Ing. Michal Moučka, Ph.D.

Liberec 2019

Technická univerzita v Liberci
Fakulta strojní
Akademický rok: **2017/2018**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jakub Bláha**
Osobní číslo:	**S16000311**
Studijní program:	**N2301 Strojní inženýrství**
Studijní obor:	**Výrobní systémy a procesy**
Název tématu:	**Implementace knihovny funkcí pro grafický displej**
Zadávající katedra:	**Katedra výrobních systémů a automatizace**

Zásady pro vypracování:

1. Prostudujte základní algoritmy rastrové grafiky s využitím doporučené literatury.
2. Seznamte se s možnostmi grafického displeje RAYSTAR RG320240B-B/W-V a jednočipového mikropočítače PIC24F.
3. Naprogramujte v jazyku C knihovnu funkcí pro mikropočítač PIC24F pro práci s daným grafickým displejem. Knihovna musí umožnit ovládání displeje, zobrazení textu, úsečky a elipsy.
4. Ověřte funkčnost a správnost řešení.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **cca 45 stran + přílohy**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] HEROUT, P. Učebnice jazyka C (6. vydání). České Budějovice: KOPP nakladatelství. 2009. ISBN 978-80-7232-383-8
[2] ŽÁRA, J. a kol. Počítačová grafika principy a algoritmy. Praha: GRADA Vydavatelství a nakladatelství, 1992. ISBN 80-85623-00-5.
[3] ŽÁRA, J., B. BENEŠ, P. FELKEL Moderní počítačová grafika. Praha: Computer Press, 1998. ISBN 80-7226-049-9.
[4] PIC24FV16KM204 Family Data Sheet [online]. 2017 [cit. 2017-09-25]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/30003030b.pdf
[5] RAiO RA8835 Dot matrix LCD Controller Specification [online]. 2017 [cit. 2017-09-25]. Dostupné z: http://www.pacificdisplay.com/ics_app%20notes/raio/RA8835_DS_v22.pdf
[6] RG320240B-BIW-V [online]. 2017 [cit. 2017-09-25]. Dostupné z: http://www.tme.eu/en/Document/857c24a567133be9bf0b92b9a8926454/RG320240B-BIW-V.pdf

Vedoucí diplomové práce: **Ing. Michal Moučka, Ph.D.**
Katedra výrobních systémů a automatizace

Datum zadání diplomové práce: **15. listopadu 2017**

Termín odevzdání diplomové práce: **15. května 2019**

L.S.

prof. Dr. Ing. Petr Lenfeld
děkan

Ing. Petr Zelený, Ph.D.
vedoucí katedry

V Liberci dne 15. listopadu 2017

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že texty tištěné verze práce a elektronické verze práce vložené do IS STAG se shodují.

2. 5. 2019 Bc. Jakub Bláha

Poděkování

Chtěl bych poděkovat všem, kteří mě podporovali při tvorbě diplomové práce. Především panu Ing. Michalovi Moučkovi, Ph.D. za vedení diplomové práce, za cenné rady, předmětné připomínky a trpělivost. Rád bych chtěl poděkovat mé rodině, mamce a tátovi a své partnerce Kristýně Langové za podporu a trpělivost, která mi byla poskytována během celého studia.

Implementace knihovny funkcí pro grafický displej

ANOTACE:

Diplomová práce se zabývá tvorbou knihovny funkcí pro grafický displej. Práce shrnuje základní algoritmy počítačové grafiky s využitím doporučené literatury. V rámci této práce jsem měl příležitost seznámit se s možnostmi grafického displeje a jednočipového mikropočítače. V teoretické části jsou nastíněny jednotlivé algoritmy rastrové grafiky a následně jsou popsány hlavní prvky obvodu. V praktické části je navrženo propojení mikropočítače s grafickým displejem a ověření správnosti zapojení. Práce se zaměřuje na naprogramování grafických funkcí v jazyce C pro mikropočítač PIC24F s daným displejem. Poté je ověřena funkčnost a správnost zapojeni celé sestavy.

Klíčová slova: rastrová grafika, mikropočítač PIC24, grafický displej, rastrové algoritmy, knihovna funkcí, programovací jazyk C

Implementation function library for a graphic display

ANNOTATION:

The thesis deals with creation of a library of functions for graphic display. This thesis summarizes basic computer graphics algorithms using recommended literature. Within this thesis, I had the opportunity to get acquainted with the possibilities of a graphical display and a single-chip microcomputer. In the theoretical part are outlined individual algorithms of raster graphics and then the main elements of the circuit are described. In the practical part, there is designed microcomputer connections with a graphical display and verified the correctness of connections. The thesis is focused on programming graphical functions in C language for PIC24F with given display. Then the functionality and correctness of the whole assembly are verified.

Key words: raster graphics, microcomputer PIC24, graphic display, raster algorithms, function library, programming language C

Obsah

Seznam obrázků

Seznam tabulek a grafů

Seznam termínů a zkratek

atd.	a tak dále
$\overline{CS}$	Řídící signál čipu
$\overline{RD}$	Signál pro čtení
$\overline{WR}$	Signál pro zápis
A0	Řídící signál
C	Programovací jazyk C
CAN	Sběrnice
CCFL	fluorescenční zdroje světla se studenou katodou
CSRW	Nastavení adresy kurzoru
č.	číslo
D1	Digitální signál
dsPIC30	16-bitové mikrokontroléry
dsPIC33	16-bitové mikrokontroléry
DSTN	Dvojitá vrstva STN
EasyEDA	Program pro tvorbu plošných spojů
EEPROM	Elektronicky vymazatelná paměť pouze pro čtení
FSTN	STN s kompenzačním filmem
glcd_clear_graphics	Funkce k vyčištění paměti
glcd_cmdread	Funkce pro zjištění stavu ovladače displeje
glcd_cmdwrite	Funkce pro odeslání příkazu
glcd_dataread	Funkce pro čtení dat z paměti
glcd_datawrite	Funkce pro zápis dat
glcd_init_graphics	Funkce k nastavení displeje
glcd_setpixelxy	Funkce, která vykreslí pixel
GND	Uzemnění
GPS	Navigace

HD	Vysoké rozlišení
HP laserJet M15w	Označení laserové tiskárny
HT16K33	Model 7-segmentového displeje
checkbusy	Funkce, která kontroluje stav displeje
IC	Integrovaný obvod
IDC	Druh konektorů
KB	Jednotka kapacity paměťových médií
LATA	LATCH registr portu A
LATB	LATCH registr portu B
LCD	Displej z tekutých krystalů
LCM	Modul tekutých krystalů
LED	Elektroluminiscenční dioda
MCLR	Externí reset
MCOT128128C1V	Model grafického OLED displeje
MICROCHIP	Microchip Technology Inc.
MP3	Formát kódování audia
MPLAB IDE XC16	Programocí software firmy Microchip,
MREAD	Čtení z paměti
MWRITE	Zápis do paměti
např.	na příklad
newdata	Uložení nových dat
olddata	Uložení starých dat
OLED	Elektroluminiscenční látka využívající organické materiály
Pdf	Souborový formát
PIC Microchip	Jednočipové mikropočítače vyráběné firmou
PIC24	Mikrokotroléry rodiny PIC24

PIC24F	Mikrokontroléry rodiny PIC24F
PIC24FV16KM202	Označení mikrokontroléru
PIC24FVXXKMXXX	Označení rodiny mikrokotroléru
PIC32	32-bitové mikrokontroléry
PORTA	8-bitový port A
PORTB	8-bitový port B
PPI	Body na palec
RA	Digitální signál portu A
RA8835	Označení grafického ovladače
RaiO	Firma RaiO Technology Inc.
RAM	Operační paměť
RAYSTAR OPTRONICS	Firma Raystar Optronics
RESET	Signál k resetování displeje
RG320240A	Model grafického LCD displeje
RG320240B-BIW-V	Označení grafického displeje
ROM	Paměť pouze pro čtení
SPI	Sériové periferní rozhraní
STN	Technologie výroby tekutých krystalů Super-Twisted Nematic
TFT	Thin-film transistor, technologie výroby displejů z tekutých krystalů
TN	Technologie výroby tekutých krystalů Twisted Nematic
TRIS	Registr pro nastavení směru pinů
TRISA	Registr pro nastavení směru pinů portu A
tzn.	to znamená
USART	Zařízení pro sériovou komunikaci
USB	Univerzální sériová sběrnice

Seznam symbolů

$d_1[-]$ – rozdíl vzdáleností

$d_2[-]$ – rozdíl vzdáleností

$p_i[-]$ – predikce

$p_{i+1}[-]$ – predikce následujícího kroku

$x_c[-]$ – x – ová souřadnice centrálniho bodu

$x_i[-]$ – souřadnice na ose x i – tého kroku

$x_{i+1}[-]$ – souřadnice na ose x následujícího kroku

$y_c[-]$ – y – ová souřadnice centrálního bodu

$y_i[-]$ – souřadnice na ose y i – tého kroku

$y_{i+1}[-]$ – souřadnice na ose y následujícího kroku

$\Delta d[-]$ – rozdíl vzdáleností d_1 a d_2

$\Delta x[-]$ – rozdíl x – ových souřadnic

$\Delta y[-]$ – rozdíl y – ových souřadnic

$a[-]$ – hlavní poloosa elipsy a

$b[-]$ – hlavní poloosa elipsy b, koeficient úsečky

$\mathcal{F}(x, y)$ – funkce závislá na souřadnicích x a y

$m[-]$ – směrnice úsečky

$r[-]$ – poloměr kružnice

$x[-]$ – souřadnice na ose x

$y[-]$ – souřadnice na ose y

Úvod

Diplomová práce se zabývá implementací základních algoritmů počítačové grafiky, které jsou vysvětleny na geometrických útvarech jako úsečka, kružnice a elipsa. Algoritmy pro jednotlivé případy jsou vysvětleny a popsány v kapitole č. 3.2. V této kapitole je také nastíněno rastrové vykreslování písma. Jakmile byly jednotlivé algoritmy vysvětleny, bylo nutné definovat základní funkce mikropočítače, využití a možnosti tohoto zařízení. S tím souviselo prostudování příslušných dokumentů k mikropočítači poskytnutých firmou MICROCHIP. Dále došlo k seznámení se s uživatelským prostředím programu MPLAB IDE XC16. Veškeré tyto informace jsou shrnuty v kapitole č. 4. Dále jsou uvedeny základní parametry námi zvoleného grafického displeje firmy RAYSTAR OPTRONICS v kapitole č. 3.5. Tento displej je řízen grafickým ovladačem, ten je podrobně popsán v kapitole č. 3.5.1. Poté byla zmíněna správa paměti grafického displeje a její rozdělení, jelikož paměť se skládá ze dvou vrstev viz kapitola 3.5.2. S tím souvisí taktovací diagram, který slouží k zadávání příkazů, zápisu dat a jejich čtení. Tato problematika je popsána v kapitole č. 3.5.3.

Na základě informací získaných z dokumentů výrobců bylo možné vytvořit paralelní komunikaci mezi mikročipem a grafickým ovladačem viz kapitola č. 3.5.4.

V kapitole č. 4.1 jsou popsány jednotlivé funkce celého programu a podrobně vysvětleny s komentářem autora. Popsány jsou vstupní parametry, účel funkce a případně výstup.

V praktické části bylo nutné zprvu ověřit správnou funkčnost mikročipu, poté bylo vytvořeno schéma zapojení a postupně byly propojeny jednotlivé komponenty. Později, na základě získaných informací z teoretické části, byla vytvořena paralelní komunikace mezi displejem a mikročipem. Tímto úkonem skončila hardwarová část a započala softwarová.

Následně byla otestována správnost a odhalily se disfunkce zapojení. Jakmile bylo možné displej inicializovat, začaly se testovat jednotlivé funkce algoritmů s průběžným zaznamenáváním možných problémů a jejich odstranění. Výsledkem bylo správné zobrazení na displeji jednotlivých funkcí v kapitole č. 4.6.

Ověření správnosti jednotlivých funkcí bylo provedeno vlastním spuštěním funkce a zkontrolování výsledku na displeji viz kapitola č. 4.5.

Na závěr byly zhodnoceny výsledky diplomové práce a návrh na možné zlepšení viz kapitola č. 5.

1 Cíle Práce

Cílem této diplomové práce bylo naprogramovat základní algoritmy počítačové grafiky. Seznámení se s mikropočítačem a displejem obnášelo prostudování veškerých příslušných dokumentů poskytnutých výrobcem pro jednotlivé komponenty. Došlo k realizování schéma zapojení. Byly ověřeny funkčnost a správnost zapojení displeje a bylo vytvořeno komunikační rozhraní mezi mikrokontrolérem a displejem.

Program by měl umožňovat vykreslení textu, úsečky a elipsy na displeji pomocí algoritmů rastrové grafiky, které byly uvedeny v doporučené literatuře. Proto byla pro každou metodu vytvořena funkce v jazyce C s potřebnými vstupy.

Výsledky funkcí rastrové grafiky bylo nutné ověřit, zdali byl výsledek správný a bez defektů.

K úspěšnému splnění výše zmíněných cílů diplomové práce bylo nutné využití znalostí a principů aplikované kybernetiky, a navíc bylo potřebné znát detailně programovací jazyk C.

2 Teoretická část

2.1 Počítačová grafika

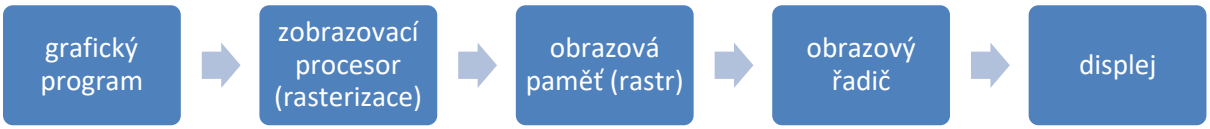
Uplatnění a využití počítačové grafiky jsou mnoha. Pomocí počítače a displejů lze snadno a rychle vytvářet jednoduchou tvorbu obrázků. Grafický displeje nám přináší výhody skoro ve všech oblastech. Proto nacházíme prvky počítačové grafiky ve většině počítačových aplikací. Grafická zařízení se v dnešní době běžně využívají v oblastech jako jsou obchod, průmysl, řízení, umění, hry, reklamy, výuka, věda, a výzkum, výcviky a školení, lékařství. [1]

Základní grafické zařízení je displej. Je určen k zobrazování grafických údajů. V současnosti jsou nejvíce využívané rastrové displeje. Další zařízení využívající tohoto systému jsou souřadnicové zapisovače, resp. plotr nebo tiskárny. Pro zadávání dat do těchto systémů slouží různá zařízení jako jsou myš, klávesnice nebo snímače souřadnic. [1]

V současné době je nejvíce rozšířena technologie LCD. Tato technologie využívá svítivé tekuté krystaly. Uplatňuje se v malých i velkých displejích a přenosných počítačích. Aby bylo možné zobrazit obrázek na displeji, je nutné ho nejprve uložit v pomocné paměti, kterou displej opakovaně prochází. Následně dojde k polarizaci příslušných tekutých krystalů umístěných před odrazivou plochou a přestanou propouštět světlo. Na displeji se objeví množina tmavých nebo světlých bodů. [1]

Rastrové systémy neobsahují grafické příkazy, ale v paměti jsou uloženy informace o velikosti jasu jednotlivých bodů na obrazovce. Tuto paměť označujeme jako obrazovou nebo je označována pod názvem bitová mapa. Paměť je opakovaně čtena s určitou frekvencí a přečtená data jsou převedena do obrazové podoby. Tento proces vykonává obrazový řadič. [1]

Informace jsou postupně zapisovány z aplikačního programu do paměti. Tento proces se nazývá rasterizace nebo bodový rozklad. Rasterizace se provádí pomocí programu algoritmů rastrové grafiky. Případně je prováděna specializovaným grafickým procesorem, který vytváří bodový rozklad nejjednodušších grafických prvků např.: úsečky, kruhové oblouky a elipsy. [1]



Obrázek 1 – Proces vykreslování [1]

Množství potřebné paměti musí být přizpůsobeno velikosti displeje, tzn. pokud má displej rozlišení 320 x 240 pixelů, tak by paměť displeje měla být minimálně 9600 bajtů pro grafickou vrstvu, a navíc pro textovou vrstvu by mělo připadnout dalších 1200 bajtů. Používají se systémy s mnohem vyšším rozlišením a vyšší kapacitou grafické paměti. Tu lze ještě rozdělit na více částí (stránky) a ty obsahují zobrazované pixely. Do dalších částí si lze načíst data, předpřipravit si vrstvu a poté mezi nimi rychle přepínat. Grafické systémy lze charakterizovat dvěma údaji, a to rozlišovací schopnost monitoru a velikost obrazové paměti. [1]

2.2 Základní algoritmy rastrové grafiky

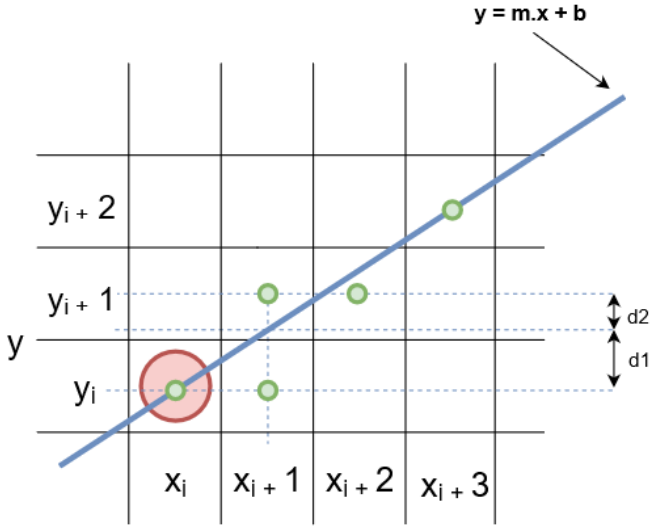
V této kapitole jsou vysvětleny základní algoritmy počítačové grafiky, které byly použity v diplomové práci.

2.2.1 Generování úsečky pomocí Bresenhamova algoritmu

Tento algoritmus generuje efektivně jednotlivé body na úsečce. Hledá body, které leží nejblíže původní úsečce pomocí celočíselné aritmetiky. Na obrázku č. 1 je zobrazena část displeje, ve které mají být vygenerovány jednotlivé pixely. Jakmile je vykreslen první bod levého konce úsečky, je nutno určit, jestli vykreslit další pixel se stejnou souřadnicí y nebo přičíst k y-ové souřadnici jedničku. Volíme pixel, který je nejblíže původní hodnotě y na úsečce. [1]

Tento algoritmus lze provést v několika postupech. Úsečka s kladnou směrnicí má řídící osu x. To znamená, že ke každé souřadnici x, ke které připočteme jedničku, dopočítáme novou souřadnici y nejblíže skutečné hodnotě y. Přepokládáme, že výchozí pixel se souřadnicemi (x_i, y_i) byl již zobrazen. Následně se rozhoduje o dalším pixelu, který má dvě možné pozice (x_{i+1}, y_i) nebo (x_{i+1}, y_{i+1}). Jak je patrné z obrázku č. 2,

rozhodují rozdíly vzdálenosti pixelů na hodnotách y_i a y_{i+1} se skutečnou hodnotou na úsečce. Tyto vzdálenosti jsou označeny jako d_1 a d_2. [1]



Obrázek 2 – Tvorba úsečky [3]

Poté lze hodnotu y vypočítat takto:

$$y = m.(x_i + 1) + b \tag{1}$$

Velikosti d_1 a d_2 jsou vypočítány z následujících rovnic:

$$d_1 = y - y_i = m.(x_i + 1) + b - y_i \tag{2}$$

$$d_2 = y_i + 1 - y = y_i + 1 - m.(x_i + 1) - b \tag{3}$$

Rozdíl Δd je pak:

$$\Delta d = d_1 - d_2 = 2.m.(x_i + 1) - 2y_i + 2b - 1 \tag{4}$$

Podle Δd lze snadno určit, který bod je blíže skutečné úsečce. Pokud je hodnota Δd záporná, pixel y_i je blíže. V opačném případě (kladná hodnota Δd) je pixel y_{i+1} blíže. Není důležitá hodnota Δd, ale její znaménko. Ještě je nutné převést rovnici do celočíselné aritmetiky tím, že jí vynásobíme Δx. [1]

$$p_i = \Delta d.\Delta x = 2.\Delta y.x_i - 2.\Delta x_i.y_i + 2\Delta y + \Delta x.(2b - 1) \tag{5}$$

V této rovnici pak označíme $2\Delta y + \Delta x(2b - 1)$ jako konstantu a následně je vyloučena po dalších úpravách. [1]

$$p_{i+1} = 2.\Delta y.x_{i+1} - 2.\Delta x.y_{i+1} + konst. \tag{6}$$

Po odečtení rovnic, získáme

$$p_{i+1} = p_i + 2.\Delta y - 2.\Delta x + (y_{i+1} - y_i) \quad (7)$$

Proměnou hodnotu p_i, neboli predikci, lze snadno získat z předcházející predikce pomocí následujících rovnic. [1]

Pro $p_i < 0$ *platí*:

$$p_{i+1} = p_i + 2.\Delta y \quad (8)$$

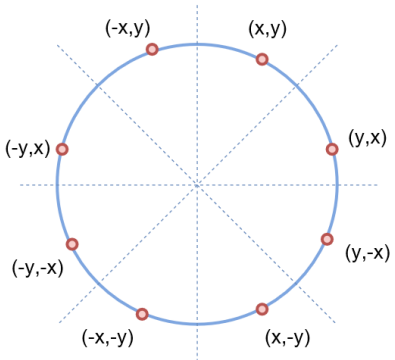
Pro $p_i \geq 0$ *platí*:

$$p_{i+1} = p_i + 2.\Delta y - 2.\Delta x \quad (9)$$

První hodnotu predikce vypočítáme $p_1 = 2.\Delta y - \Delta x$ pomocí rovnice 7 a použijeme počáteční souřadnice (x_1, y_1). Predikce nám určuje, jaký pixel je vybrán. Pokud je $p_i < 0$, je vybrána souřadnice y pixelu nebo naopak $p_i \geq 0$, vybereme pixel, který má souřadnici y větší o jedničku. Se znalostí tohoto algoritmu je možné vytvořit program v jazyce C, který nám umožní vygenerovat libovolnou úsečku. [1]

2.2.2 Generování kružnice pomocí Bresenhamova algoritmu

Stejně jako u algoritmu pro tvorbu úsečky se bere v úvahu počáteční bod, a poté se generují ostatní pixely pomocí celočíselné aritmetiky. Při každém kroku se rozhoduje, který pixel je blíže skutečné kružnici s tím, že počátek souřadné soustavy je v $(x_c = 0, y_c = 0)$. Počítají se body pouze v jedné osmině, a pak se pomocí symetrie vykreslí zbylé body (viz obrázek č. 3). Řídící osa je x, která začíná v $x = 0$ a končí hodnotou $x = y$. Výchozí bod má souřadnice $(0, r)$. [1]



Obrázek 3 – Osy symetrie [4]

Algoritmus na generování kružnice je založen na stejném principu jako Bresenhamův algoritmus pro tvorbu úsečky. Tento algoritmus využívá celočíselnou aritmetiku a pro každý následující bod počítá predikci. Tato predikce rozhoduje, jaký bod se vykreslí. [2]

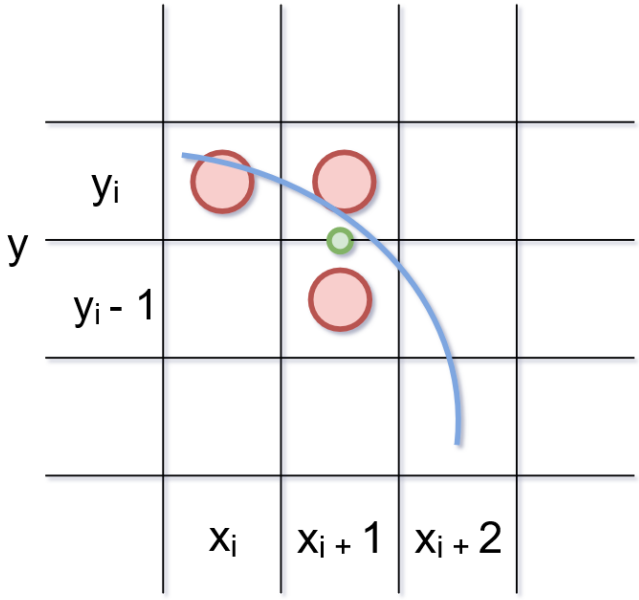
Kružnici lze vyjádřit jako funkci:

$$\mathcal{F}(x,y): x^2 + y^2 - r^2 = 0 \qquad (10)$$

Pro výpočet libovolného bodu předpokládáme, že výchozí bod (x_i, y_i) byl zvolen pomocí predikce. Následující pixel může mít dvě možné souřadnice $(x_i + 1, y_i)$ nebo $(x_i + 1, y_i - 1)$. Dosazením do funkce č. 10 $\left(x_i + 1, y_i - \frac{1}{2}\right)$ se určí znaménko predikce. Podle výsledku predikce je zjištěno, zdali se bod nachází uvnitř kružnice nebo na vnější straně viz rovnice č. 11. [1]

$$p_i = \mathcal{F}\left(x_i + 1, y_i - \frac{1}{2}\right) = (x_i + 1)^2 + (y_i - \frac{1}{2})^2 - r^2 = 0 \qquad (11)$$

Když vyjde znaménko predikce záporné, tak se vykreslí pixel se souřadnicí y_i. Pokud je znaménko kladné, bude vykreslen bod se souřadnicí $y_i - 1$. Příklad části kružnice je uveden na obrázku č. 4. [1]



Obrázek 4 – Tvorba úsečky [5]

V dalším kroku se zjišťuje následující hodnota predikce p_{i+1} , ta se určí z předcházející predikce a vypočítá se z rovnice č. 12. [1]

$$p_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - \frac{1}{2})^2 - r^2 = 0 \qquad (12)$$

Po dosazení $x_{i+1} = x_i + 1$ do rovnice č. 12 a následným odečtením od rovnice č. 11, platí následující rovnice č. 13.[1]

$$p_{i+1} = p_i + 2.x_i + 3 + (y_i - \frac{1}{2})^2 - (y_{i+1} - \frac{1}{2})^2 \qquad (13)$$

Pokud je hodnota predikce záporná, tak souřadnice y se nemění. V opačném případě, kdy predikce je kladná, souřadnice y se sníží o jedna. Pro predikci platí následující rovnice č. 14 a 15. [1]

Pro $p_i < 0$ *platí*:

$$p_{i+1} = p_i + 2.x_i + 3 \qquad (14)$$

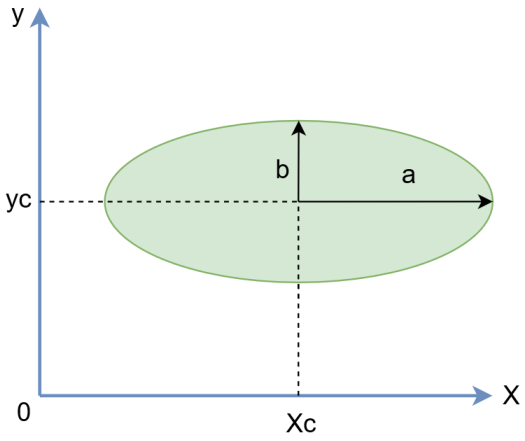
Pro $p_i \geq 0$ *platí*:

$$p_{i+1} = p_i + 2.x_i + 5 - 2.y_i \qquad (15)$$

Rovnice č. 14 a 15 umožňují vygenerovat jednotlivé body na kružnici. Všechny tyto vztahy jsou použity při tvorbě programu v jazyce C, který bude vykreslovat libovolné kružnice.

2.2.3 Generování elipsy pomocí Bresenhamova algoritmu

Tento algoritmus vychází z rovnice elipsy, která je definovaná bodem (x_c, y_c) (viz obrázek č. 5). Elipsa má dvě hlavní poloosy, které jsou rovnoběžné se souřadným systémem a jsou popsány rovnicí č. 15. [1]



Obrázek 5 – Elipsa [6]

$$\frac{(x-x_c)^2}{a^2}+\frac{(y-y_c)^2}{b^2}=1 \tag{15}$$

Výpočet jednotlivý bodů elipsy je nejvýhodnější počítat v počátku souřadnicového systému. Pak jednotlivé body posunout do požadovaného místa o vektor posunutí (x_c, y_c). Elipsu se středem v nule, lze vyjádřit následující funkcí č. 16. [1]

$$\mathcal{F}(x,y): b^2x^2+a^2y^2-a^2b^2=0 \tag{16}$$

Generování bodů elipsy se provádí obdobným způsobem jako u algoritmu pro kružnici. Jelikož má elipsa dvě osy symetrie, lze pro jeden bod dopočítat čtyři pixely na elipse. Pro výpočet predikce elipsy s řídící osou x platí následující rovnice č. 17 a 18. [1]

Pro $p_i < 0$ *platí*:

$$p_{i+1}=p_i+b^2(2.x_i+1) \tag{17}$$

Pro $p_i \geq 0$ *platí*:

$$p_{i+1}=p_i+b^2(2.x_i+1)-2.a^2.y_i \tag{18}$$

První hodnotu predikce p_i vypočítáme z rovnice č. 19. V tomto vztahu dochází k celočíselnému dělení. Když je hodnota a lichá, může dojít k malé chybě. Tato chyba je zanedbatelná. [1]

$$p_i=b^2-b.a^2-\frac{a^2}{4} \tag{19}$$

Všechny tyto vzorce jsou použity k tvorbě algoritmu, který je naprogramován v jazyce C. Program umožňuje vykreslit libovolnou elipsu na displeji.

2.3 Zobrazovací zařízení

Zobrazovací zařízení jsou výstupní zařízení, která můžou být připojena k vstupním nebo výstupním portům mikrokontroléru. Většina elektronických zařízení, ať již spotřebitelských, komerčních nebo průmyslových, mají nějakou formu zobrazovacího zařízení např.: mobilní telefony, kalkulačky, GPS systémy, tiskárny, počítače, MP3 přehrávače, mikrovlnná trouba a tak dále. [7]

V této kapitole jsou popsána pouze malá zobrazovací zařízení, která jsou použita v projektech založených na mikrokontrolérech. Obecně se tato zařízení dělí na tři základní skupiny: LED, OLED a LCD. [7]

2.3.1 LED

Displeje založené na světelných diodách – LED (viz obrázek č. 6) jsou dále rozděleny do dvou skupin: jednoduché LED nebo 7 - segmentové LED. Jednoduchá LED zařízení se skládají z jedné diody nebo pole diod. Běžně se používají k indikaci nějakého statusu např.: zapnutí, vypnutí elektrického zařízení, zvolený režim atd. LED diody jsou k dispozici v různých barvách (červená, zelená, bílá atd.). Jsou napřímo připojeny ke vstupům/výstupům jednotlivých portů mikrokontrolérů s rezistorem, který omezuje procházející proud. [7]

Obrázek 6 – Barevné LED diody [8]

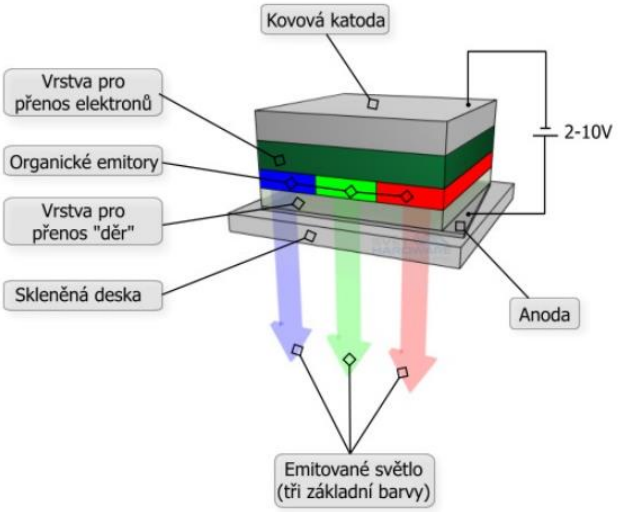
2.3.2 7 - segmentová LED

Tato zařízení se používají k zobrazování numerických dat. Čísla jsou vytvořena ze sedmi segmentů viz obrázek č. 7. Požadované číslo se zobrazí zapnutím a vypnutím konkrétních segmentů. Jsou dva typy sedmi-segmentových displejů. V běžných anodových displejích je anoda připojena ke zdroji napětí a individuální segmenty jsou zapínány jejich uzemněním. Dále je běžný katodový typ displeje. Katoda je připojena k zemi a individuální segmenty jsou zapínány tím, že se přivede napětí k požadovaným segmentům. Oba tyto typy displejů lze snadno připojit ke vstupům a výstupům mikrokontroléru. K vyobrazení číslic v rozsahu nula až devět je použita jedna číslice. Pokud je požadováno zobrazení vyššího řádu, je zapotřebí použít více číslic. Při aplikaci vyššího počtu číslic je každé číslo zapínáno a vypínáno řízením příslušných pinů. Jsou střídavě aktivovány a deaktivovány tak rychle, že z pohledu uživatele je vidět stacionární číslo. [7]

Obrázek 7 - 7 - segment LED HT16K33 Backpack [9]

2.3.3 OLED

Tato technologie je založena na elektroluminiscenci v organických materiálech. Jsou konstruovány tak, že organický materiál je mezi anodou a katodou. Při průchodu elektrického proudu začne organický materiál produkovat jasné elektroluminiscenční světlo. Princip této technologie je znázorněn na obrázku č. 8. Rozlišují se dva typy OLED displejů v závislosti na použitém materiálu. Mohou být buď na bázi molekul, anebo na bázi polymerů. Pomocí OLED displejů lze zobrazovat text a grafické obrázky viz obrázek č. 9. Lze je využívat bez podsvícení, mohou být použity ve venkovním prostoru, ale i v interiéru při nízkém okolním osvětlení. [7]



Obrázek 8 - Princip OLED displejů [10]

Obrázek 9 - Grafický OLED displej MCOT128128C1V [11]

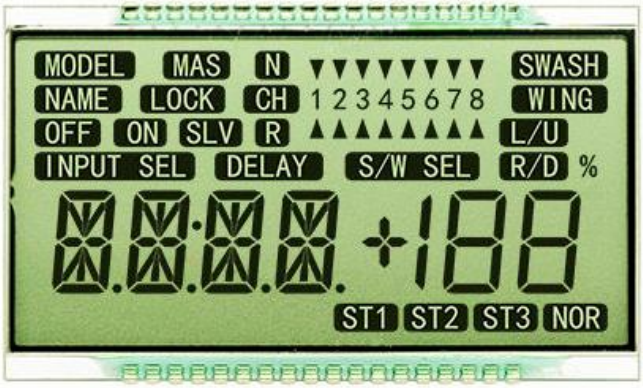
OLED displeje mají několik výhod oproti konkurenčním technologiím. Mají velký pozorovací úhel a vylepšený jas. Barvy jednotlivých pixelů zobrazují správnou barvu při změně pozorovacího úhlu. Mají velmi rychlou dobu odezvy až 200krát rychlejší než LCD displeje. Obraz těchto displejů je velmi jasný a ostrý. Výsledný produkt může být lehký a extrémně tenký. Spotřeba energie je také nízká. [7]

Tato technologie má i některé nevýhody. Vlastní výroba těchto displejů je nákladná. Životnost OLED zařízení je omezená, obvykle bývá 14000 hodin, což odpovídá životnosti pěti let osm hodin denního používání. Při kontaktu s vodou mohou být poškozeny, nicméně to lze odstranit utěsněním rámečku displeje na úkor vyšších nákladů. Další problém těchto displejů je při zobrazování stejného obsahu, kde může dojít k vypálení pixelů, což zapříčiní jejich vyblednutí. Při vystavení velmi vysokému UV záření může dojít k poškození displeje. Aby se tomu zabránilo, výrobci obvykle přidávají ochranný filtr. Materiál k zobrazení modré barvy degraduje rychleji než ostatní barvy. Proto se postupem času barvy zobrazují špatně. [7]

2.3.4 LCD

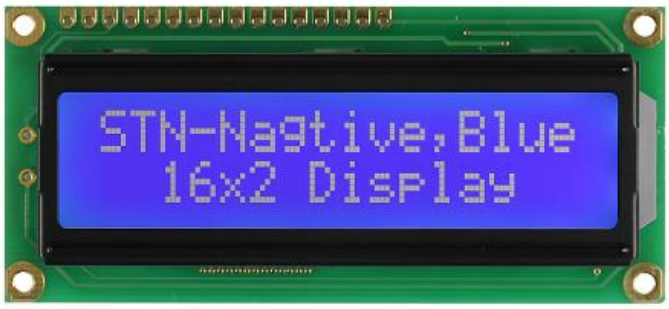
Displej z tekutých krystalů je nejběžněji používaný displej současnosti. Existují tři základní typy tohoto druhu displejů segmentový LCD, maticový LCD a grafický LCD. [7]

Segmentový displej se také nazývá alfanumerické LCD viz obrázek č. 10. Tento displej může zobrazovat čísla, která jsou reprezentovány sedmi segmenty. Nebo zobrazuje čísla a římská písmena reprezentovány 16 segmenty. Lze zobrazit také symboly. Segmentové displeje jsou omezeny ve vykreslování čísel, textu a symbolů. Pokud je potřeba zobrazit jiný znak nebo nějaký odlišný tvar, měl by se použít buď maticový, nebo grafický displej. [7]



Obrázek 10 - 16 - segmentový LCD displej [12]

Maticový displej je také znám jako znakový LCD. Nejběžněji používaný displej je dvouřádkový s 16 znaky. Každý znak reprezentuje 5 x 7 pixelů. Toto zařízení může zobrazovat alfanumerická data včetně symbolů. Typický displej je znázorněn na obrázku č. 11. [7]



Obrázek 11 - STN LCD Display 16x2 [13]

Grafické LCD displeje jsou tvořeny pixely, a to umožňuje uživateli největší flexibilitu. Jednotlivé pixely jsou uspořádány v řádcích a sloupcích. Každý pixel může být ovládán individuálně. Tyto displeje se používají k zobrazení čísel, písmen, symbolů, složitějších tvarů nebo obrázků. Na obrázku č. 12 je grafický displej značky Raystar. [7]

Obrázek 12 - Grafický LCD Display RG320240A [14]

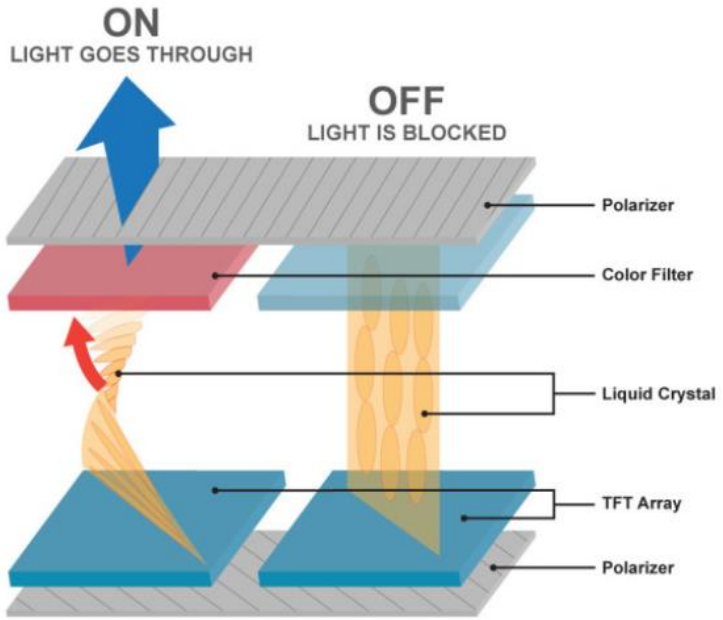
Všechny tyto displeje neprodukují vlastní světlo, proto je vyžadováno použití externího zdroje světla, aby byl displej čitelný. Je tedy nutné vložit za LCD panel zdroj světelného záření (plochá nízkotlaká výbojka). [7]

Lze realizovat podsvícení, které je barevné. Toho může být docíleno pomocí:

- LED diody,
- elektroluminiscenční fólie,
- fluorescenční lampy,
- chladné katody (CCFL). [16]

Princip LCD displejů spočívá ve využívání vlastností tekutých krystalů, které modifikují světlo. Ve standardních LCD displejích se nachází vrstva molekul, která je uspořádána mezi dvěma elektrodami z oxidu cínu a mezi dvěma polarizačními filtry, které jsou umístěny s osou polarizace pootočenou o 90 stupňů viz obrázek č. 13. Světlo prochází LCD displejem předním polarizačním filtrem. Projde tekutými krystaly, které pootočí procházející světlo. Rotace světla bývá obvykle 90 stupňů ve většině typů

LCD displejů. Při vypnutém stavu se světlo otočí a projde i druhým polarizačním filtrem. Při průchodu napětí elektrodami získáme stav zapnutí. Orientace tekutých krystalů se změní tak, že jsou paralelně k elektrickému poli a kroucená struktura tekutých krystalů se narovná. Světlo není nadále otáčeno a prochází k druhému polarizačnímu filtru, kde je absorbováno. To zapříčiní, že pixel displeje vypadá tmavě. [7]



Obrázek 13 - Princip technologie Twisted Nematic - TN [15]

LCD displeje mohou být klasifikovány jako displeje s pasivní a aktivní maticí. Záleží na použitém schématu adresování jednotlivých pixelů. Pixely jsou adresovány řádky a sloupci. V pasivním displeji jsou transistory použity k aktivování řádků a sloupců, ale ne pro každý pixel. Na druhou stranu u aktivních displejů jsou tranzistory použity pro každý červený, modrý a zelený pixel, aby svítily v požadované intenzitě. Pasivní maticové displeje jsou levnější, mají užší úhel viditelnosti než aktivní maticové displeje. Aktivní maticové displeje mají ostřejší obraz, větší kontrast a rychlejší čas odezvy. Existuje mnoho druhů LCD displejů, závisí na množství a druhu kroucení použitých tekutých krystalů. Například technologie TN (Twisted Nematic), která pootáčí světlo o -90°, dále STN (Super-twisted Nematic) s úhlem otáčení -270°, FSTN (Film Compenasated STN) a DSTN (Double Layer STN). Technologii FSTN a DSTN jsou rozšířením na technologie STN, jak již názvy napovídají. [7]

Nejrozšířenější technologií v současné době jsou TFT LCD displeje. Jedná se o technologii, která využívá tenkého tranzistorového filmu. Nejvíce se využívá

u mobilních telefonů, monitorů notebooků a monitorů stolních počítačů. TFT displeje využívají aktivní matici a produkují jedno z nejlepších rozlišení. Vysoké rozlišení zásadně zvyšuje cenu displeje. Tranzistory tohoto displeje jsou vyrobeny z tenkého filmu amorfního křemíku, který je uložen na skleněném panelu. Výhody těchto displejů jsou vynikající doba odezvy a ostrost obrazu. Některé TFT displeje mohou mít dotykovou obrazovku, to umožňuje uživateli výběr pomocí dotknutí se příslušného místa na obrazovce. [7]

Displeje mají tři režimy zobrazení:

- Reflexní LCD displeje využívají okolního osvětlení, aby pozorovatel mohl sledovat, co je na obrazovce. Nevyužívá se optického záření, ale denního světla. Paprsky světla, které dopadají na displej, se následně odrazí zrcadlem umístěným za panelem displeje. Výhodou tohoto režimu je velmi nízká spotřeba a nenáročnost na externí zdroje. [16]
- Transmisní LCD displeje mají zabudovaný zdroj osvětlení za displejem, aby zobrazený text byl zřetelný pro pozorovatele. Zadní část tohoto displeje je průhledná, jelikož světlo musí projít displejem. Nejlepší pracovní podmínky těchto displejů jsou při horším osvětlení. [16]
- Transflexní LCD displeje jsou kombinací výše zmíněných dvou režimů. To znamená, že lze zapínat a vypínat externí zdroj osvětlení v zadní části displeje podle potřeby. Nebo využít dobrého okolního světla, a tím snížit spotřebu. [16]

Klíčové vlastnosti LCD displejů jsou:

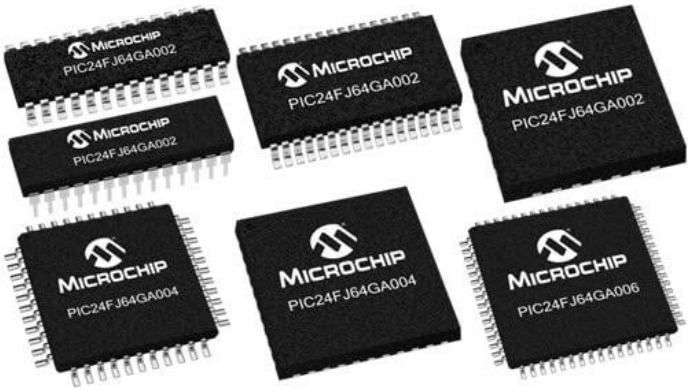
- Rozlišení
- Velikost obrazovky
- Rozteč bodů
- Doba odezvy
- Zorný úhel
- Jas
- Kontrast
- Poměr stran [7]
- Rozlišení je počet pixelů, který se měří horizontálně a vertikálně např. 1920 x 1080 (Full HD), 1280 x 720 (HD).

- Velikost obrazovky je úhlopříčka LCD displeje.
- Rozteč bodů je vzdálenost mezi dvěma sousedními stejně barevnými pixely. Rozteč se udává jako počet bodů na palec (PPI – pixels per inch). Čím menší je rozteč jednotlivých pixelů, tím je obraz ostřejší.
- Doba odezvy je minimální čas, který je zapotřebí ke změně jasu nebo barvy pixelu. Je měřena v milisekundách a obvyklé hodnoty bývají v rozsahu několika milisekund. Je žádoucí, aby tato hodnota byla co nejnižší.
- Zorný úhel je úhel pohledu na LCD displej, aniž by obraz měl jakoukoliv ztrátu detailů.
- Jas udává, kolik světla vyzařují LCD displeje.
- Poměr kontrastu je určen nejjasnější barvou (bílé) k nejtmavší barvě (černá). Vysoký kontrastní poměr je žádoucí a lze zobrazit libovolný obraz.
- Poměr stran je poměr šířky displeje k jeho výšce např. 16:9, 4:3. [7]

K tomu, aby displeje mohly vykreslovat data, potřebují mikrokontroléry, které zajišťují správu dat a komunikaci s řadičem displejů. Tato problematika je popsána v následujících kapitolách 3.4 a 3.5.

2.4 PIC Mikrokontroléry

PIC je rodina mikrokontrolérů architektury Harvard (až na výjimku 32bitových zařízení) vyráběných firmou Microchip Technology Inc. Mikroprocesory jsou k dispozici ve více jak 1000 modelech viz obrázek č. 14. Podle šířky datového slova lze mikrokontroléry třídit do tří základních skupin: 8bitové, 16bitové a 32bitové. [7]



Obrázek 14 – Mikrokontroléry rodiny PIC24[20]

Série PIC 10, 12 a 16 jsou nižší třídou, a proto se jedná o levnější produkty 8bitových mikrokontrolérů. Jsou charakterizovány nízkou rychlostí, malým počtem pinů,

nízkou cenou, malou pamětí, mají pouze 35 instrukcí, což je dělá velmi snadné k naučení. Série PIC 18 je spíš střední třída mezi 8bitovými mikrokontroléry se střední rychlostí, vyšším počtem pinů, vetší pamětí a mají přes 80 instrukcí. Tyto mikrokontroléry zahrnují různé moduly, které jsou součástí čipu, např.: CAN, USB, SPI, USART. Tato série se v současné době využívá ve většině komplexních projektů, které jsou založené na PIC mikrokontrolérů. [7]

Vyšší třídu tvoří série PIC24, dsPIC30 a dsPIC33. Jsou to 16bitové mikrokontroléry s vysokou rychlostí paměti a podporou pro časově přesné aplikace, kde je velmi důležité zpracování v reálném čase. Nacházejí uplatnění v aplikacích se zpracováním digitálního signálu a ve vysokorychlostním automatickém digitálním řízení systémů. Architektura těchto 16bitových mikrokontrolérů se liší od 8bitových, protože jsou konfigurovány pro vysokorychlostní zpracování digitálních signálů, které mají rychlé násobení a doplňkové moduly. [7]

Nejvyšší a zároveň nejnovější řadou jsou mikrokontroléry PIC32. Jedná se o 32bitové procesory s architekturou Von Neumann, které májí velkou paměť a periferní podporu. Nabízí vysokorychlostní zpracování u vysoce přesných aplikací např.: konektivita, ovládání motorů, grafika, šifrování dat atd. [7]

Pro tuto diplomovou práci byl použit mikrokontrolér série PIC24F, který splňuje požadavky grafických displejů a poskytuje požadovanou rychlost, dostatek programové paměti a velký počet vstupních a výstupních funkcí.

Mikrokontroléry série PIC24 jsou dostupné v mnoha modelech od malých 14 pinových až po 121 pinové čipy, programové paměti se pohybují v rozsahu od 4KB až do 1024 KB, paměti RAM se pohybují od 512bytů až po 96KB, vstupní a výstupní piny jsou v rozsahu od 12 do 102 pinů. [17]

V následující kapitole bude popsán mikrokontrolér vyšší třídy PIC24, pod označením PIC24FV16KM202. Tento čip byl vybrán, jelikož má nízkou cenu, přestože se jedná o velmi výkonný mikrokontrolér. Má 28 pinů, což je dostatečné množství s možností přidáním dodatečných funkcí např.: senzory. [17]

2.4.1 PIC24FV16KM202

Tento mikrokontrolér patří do rodiny PIC24FVXXKMXXX. V této rodině se nachází 8 podobných mikrokontrolérů s nepatrně odlišnými vlastnostmi. V tabulce č. 1 jsou ukázány základní specifikace mikrokontrolérů v této rodině nebo v příloze č. 2. [17]

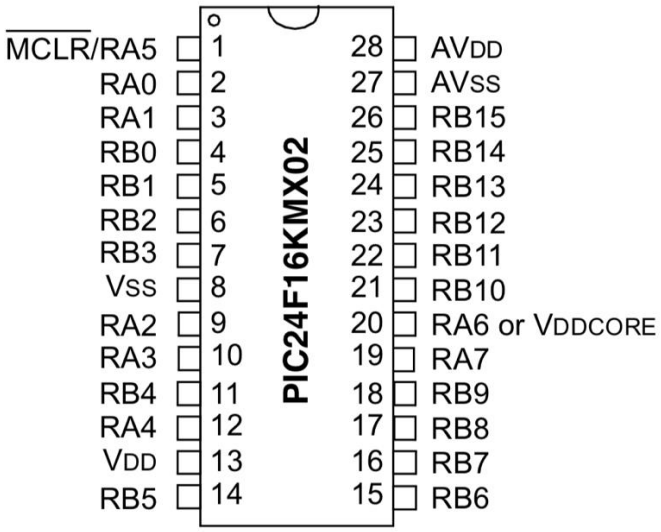
Tabulka 1 - Základní parametry mikrokontrolérů rodiny PIC24F [17]

Device	Pins	Memory			Voltage Range (V)	Peripherals											ICD BRKPT
		Flash Program (bytes)	SRAM (bytes)	EE Data (bytes)		16-Bit Timer	16-Bit MCCP/SCCP	MSSP	UART	12-Bit A/D Channels	8-Bit DAC	Op Amp	Comparators	CTMU	RTCC	CLC	
5V Devices																	
PIC24FV16KM204	44	16K	2K	512	2.0-5.5	1	3/2	2	2	22	2	2	3	Yes	Yes	2	3
PIC24FV16KM202	28	16K	2K	512	2.0-5.5	1	3/2	2	2	19	2	2	3	Yes	Yes	2	3
PIC24FV08KM204	44	8K	2K	512	2.0-5.5	1	3/2	2	2	22	2	2	3	Yes	Yes	2	3
PIC24FV08KM202	28	8K	2K	512	2.0-5.5	1	3/2	2	2	19	2	2	3	Yes	Yes	2	3
PIC24FV16KM104	44	16K	1K	512	2.0-5.5	1	1/1	1	1	22	—	—	1	Yes	—	1	3
PIC24FV16KM102	28	16K	1K	512	2.0-5.5	1	1/1	1	1	19	—	—	1	Yes	—	1	3
PIC24FV08KM102	28	8K	1K	512	2.0-5.5	1	1/1	1	1	19	—	—	1	Yes	—	1	3
PIC24FV08KM101	20	8K	1K	512	2.0-5.5	1	1/1	1	1	16	—	—	1	Yes	—	1	3

Základní specifikace mikrokontroléru PIC24FV16KM202 jsou:

- maximální operační frekvence až 32MHz,
- celkový počet pinů je 28,
- 16KB programové paměti,
- 2KB datové paměti,
- 512bytů EEPROM paměti,
- rozsah napětí 2.0 – 5.5V,
- 16bitový časovač,
- 23 vstupních/výstupních pinů. [17]

Konfigurace pinů mikrokontroléru je znázorněna na obrázku č. 15. Některé piny mohou mít více funkcí pro různé účely. Všechny funkce pinů mikrokontrolérů rodiny PIC24FVXXKMX02 jsou vypsány v příloze č. 1. Například pin 1 je označen jako $\overline{\text{MCLR}}$ a RA5, tzn. tento pin může sloužit jako reset programu (program se pustí od začátku). Pokud uživatel zvolí druhou možnost RA5, tak pin má funkci digitálního vstupu, výstupu. Aby každý pin měl správnou funkci, musí se nastavit registry mikrokontroléru. Informace pro správné nastavení lze dohledat v příslušné dokumentaci daného mikrokontroléru poskytnuté firmou Microchip. Tato problematika je popsána v kapitole 4.3. [17]



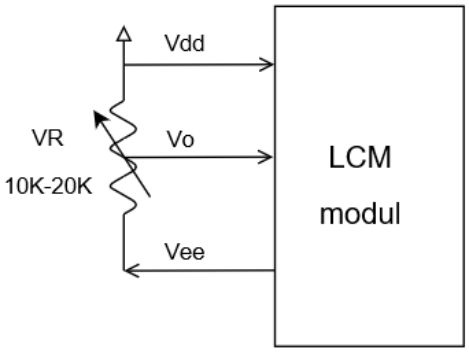
Obrázek 15 - Základní konfigurace pinů rodiny PIC24FVXXKMX02 [17]

V kapitole 3.5 jsou vysvětleny základní informace o grafickém ovladači displeje, komunikace mezi mikrokontrolérem a ovladačem viz kapitoly 3.5.2 a 3.5.3

2.5 Displej RG320240B-BIW-V

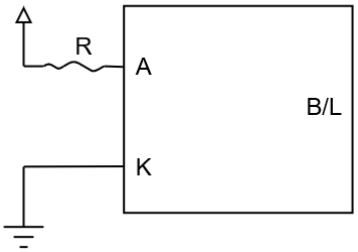
Displej pod označení RG320240B-BIW-V byl zvolen vedoucím práce. Jedná se o grafický monochromní displej vyráběný firmou Raystar Optronics. Má rozlišení 320 x 240, což je pro účely této práce dostačující. Jedná se o LCD displej, který je modrý transmisní (tento režim displeje byl vysvětlen v kapitole 3.3.4). Využívá STN technologii tekutých krystalů, která byla popsána v kapitole 3.3.4. Tento displej využívá bílou LED diodu jako podsvícení. Ovladač tohoto displeje je RA8835 firmy RaiO Technology Inc. Tyto informace jsou uvedeny v příslušné dokumentaci displeje. [18]

Na obrázku č. 16 je schéma přivedeného napětí do LCD displeje. Aby bylo možné měnit kontrast displeje, je do LCM modulu přivedeno proměnné napětí 0V – 5V, které lze měnit pomocí otočného potenciometru, resp. změnou odporu. V našem případě byl zvolen potenciometr s 20 $k\Omega$, který má funkci odporového napěťového děliče.



Obrázek 16 - Schéma zapojení podsvícení [18]

Na obrázku č. 17 je schéma zapojení LED podsvícení displeje. Podsvícení je připojeno ke zdroji napětí (anoda), které je regulováno pomocí předřadného odporu a k zemi (katoda). Jako předřadný odpor byl zvolen rezistor, který má 100Ω.



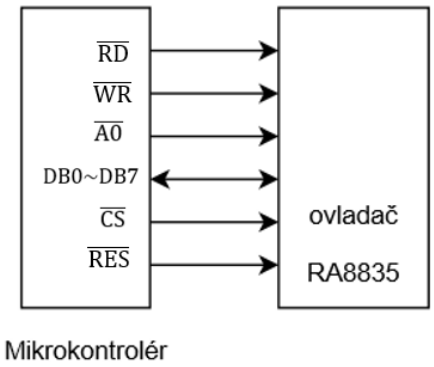
Obrázek 17 - Schéma zapojení podsvícení s [18]

2.5.1 Grafický ovladač RA8835

RA8835 je IC ovladač, který může zobrazovat text a grafiku na LCD panelu. Může zobrazovat vrstvy textu a grafiky, posouvat zobrazený obsah libovolným směrem a umí rozdělit výstup na několik obrazovek. Do paměti lze ukládat text, znakové kódy a bitmapová grafická data. Funkce ovladače displeje umožňují přenos dat z řídícího mikrokontroléru do vyrovnávací paměti, čtení dat paměti, převod dat na zobrazení pixelů a generování časových signálů pro vyrovnávací paměť. [18]

Tento ovladač má interní generátor znaků s velikostí 5x7 pixelů ve vnitřní paměti ROM. Generátor znaků podporuje až 64 znaků s velikostí 8x16 pixelů ve vnější paměti RAM a až 256 znaků s velikostí 8x16 pixelů v externím znakovém generátoru ROM. [18]

Komunikace mezi mikroprocesorem a ovladačem displeje je znázorněna na obrázku č. 18. Symboly z tohoto blokového diagramu jsou popsány v tabulce č. 2.



Obrázek 18 - Blokový diagram ovladače displeje a mikrokontroléru [18]

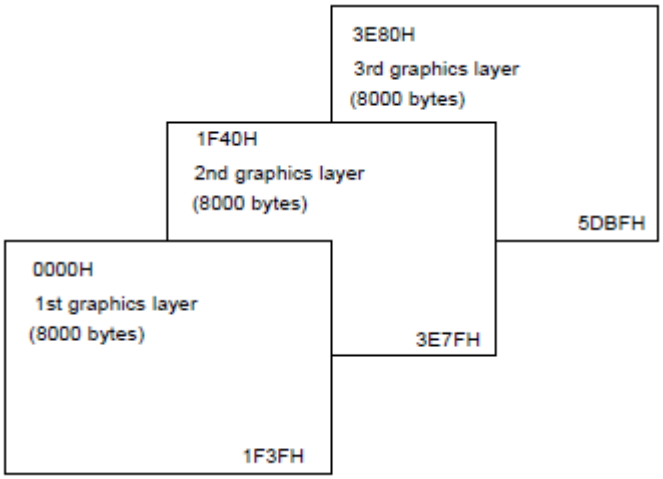
Tabulka 2 - Označení pinů displeje RG320240B-BIW-V [18]

Číslo pinu	Název	Číslo pinu	Název
1	Vss – zem 0V	11	DB4 – digitální signál 4
2	Vdd – zdroj napětí + 5V	12	DB5 – digitální signál 5
3	Vo – řídící napětí LCD	13	DB6 – digitální signál 6
4	RD – signál pro čtení	14	DB7 – digitální signál 7
5	WR – signál pro zápis	15	CS – řídící signál čipu
6	A0 – řídící signál	16	RES – signál pro resetování
7	DB0 – digitální signál 0	17	Vee – výstup záporného napětí
8	DB1 – digitální signál 1	18	FGND – uzemnění rámu
9	DB2 – digitální signál 2	19	NC – nepřipojeno
10	DB3 – digitální signál 3	20	NC – nepřipojeno

2.5.2 Správa paměti displeje

K tomu, aby bylo možné vykreslovat data na displej, musí ovladač displeje zapisovat, číst a pohybovat se kurzorem v paměti. Paměť displeje může být rozdělena na několik vrstev grafických a textových. Každá vrstva zabírá určité množství paměti a přísluší jí blok v paměti. Displej muže mít např.: tři grafické vrstvy viz obrázek č. 19,

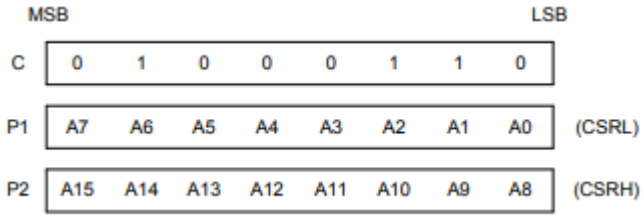
dvě textové a jednu grafickou vrstvu. Každá vrstva je v paměti definována počáteční adresou a počtem bytů, potřebným k zobrazení jedné vrstvy např.: grafická vrstva s rozlišením 320x200 spotřebuje 8000bytů v paměti viz obrázek č. 19. První vrstva začíná na adrese 0 (0000H) a končí na adrese 7999 (1F3FH), druhá vrstva začíná na adrese 8000 (1F40H) a končí na adrese 15999 (3E7FH) atd.



Obrázek 19 - Grafické vrstvy [19]

Zápis do paměti probíhá tím, že mikrokontrolér vyšle příslušnou kombinaci signálů. Jedná se o příkazové sety viz příloha č. 3 a každý příkaz má svoji unikátní kombinaci signálů viz tabulka č. 3.

Pokud chceme použít příkaz, který nastaví adresu 16bitového kurzoru, musejí se nastavit výstupní piny RD, A0, D6, D2 a D1 na vysokou hodnotu a zbytek pinů na nízkou hodnotu. Tomu odpovídá hexadecimální číslo 46. Poté následují dva byty, které mají hodnotu konkrétní adresy v paměti viz obrázek č. 20. Nejprve se pošle spodních 8bitů a pak vyšších 8 bitů. Jakmile se nastaví pozice kurzoru v paměti, lze následně použít příkazy čtení paměti (MREAD) nebo zápis do paměti (MWRITE). Aby bylo možné příkaz provést, musí se dodržet pořadí vysílání signálů, toto bude vysvětleno v kapitole č. 3.5.3.



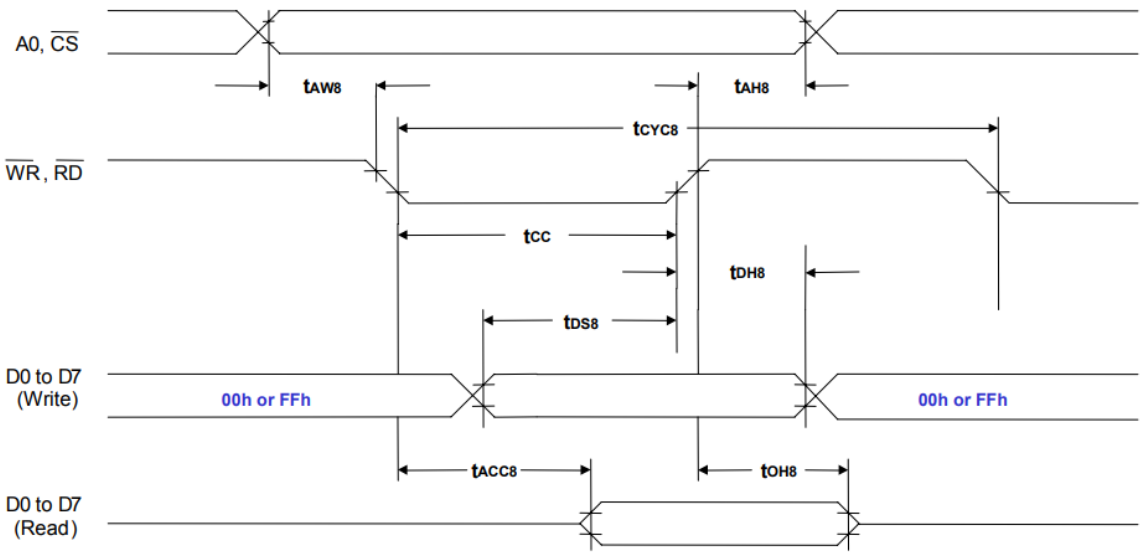
Obrázek 20 - Parametry CSRW [19]

Tabulka 3 - Příkazové sety [19]

Třída	Příkaz	RD	WR	A0	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Popis
Ovládání kreslení	CSRW	1	0	1	0	1	0	0	0	1	1	0	46	Nastaví adresu kurzoru
	CSRR	1	0	1	0	1	0	0	0	1	1	1	47	Přečte adresu kurzoru
Ovládání paměti	MWRITE	1	0	1	0	1	0	0	0	0	1	0	42	Zapsání do paměti displeje
	MREAD	1	0	1	0	1	0	0	0	0	1	1	43	Čtení z paměti displeje

2.5.3 Taktovací diagram ovladače

Na obrázku č. 21 je vyobrazen taktovací diagram rodiny 8080, kde je vysvětleno, v jakém pořadí se musí přepínat hodnoty signálů. Nejprve mikrokontrolér změní hodnoty signálů $\overline{CS}$ – řídící signál čipu a A0 – řídící signál z nízké hodnoty na vysokou nebo obráceně. Dále následuje aktivace $\overline{RD}$ – signál pro čtení nebo $\overline{WR}$ – signál pro zápis (záleží, zda se jedná o funkci čtení nebo zápis) tím, že se nastaví nízká hodnota. Poté se nastaví konkrétní hodnoty na datové sběrnici a dojde k potvrzení dat nastavením signálu $\overline{WR}$ nebo $\overline{RD}$ na vysokou hodnotu. Nakonec se přepnou hodnoty signálů $\overline{CS}$ a A0, což ukončuje příkaz. Podle taktovacího diagramu na obrázku č. 20 jsou naprogramovány funkce pro zadávaní dat a příkazů.



Obrázek 21 - Taktovací diagram rodiny 8080 [19]

2.5.4 Paralelní komunikace displeje

Paralelní komunikace probíhá pomocí pinů na mikrokontroléru a ovladači displeje. K tomuto účelu se využívají porty, které mohou být označeny PORTA, PORTB atd. Například PORTA je 8 - bitový port, který má paralelní vstupy a výstupy. Jak již bylo zmíněno, piny portu jsou vícenásobné a mohou být použity pro různé účely viz příloha č. 1. Většinu pinů portu A lze konfigurovat buď jako digitální vstupy a výstupy, nebo jako analogové vstupy.

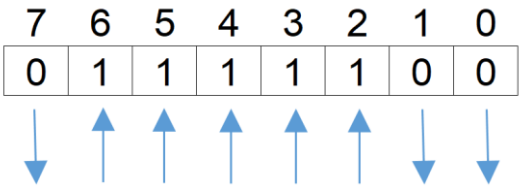
Porty lze nastavit pomocí následujících registrů:

- PORTA,
- TRISA,
- LATA.

Registr PORTA slouží k zápisu a čtení dat z portu pinů. Směr portu pinů se nastavuje pomocí TRISA registru.

Navíc ke každému PORT registru je příslušný LATCH registr. Tento registr se nazývá LATx, kde x je jméno portu, např., pro PORT A je LATA registr. Tento registr zaznamenává aktuální odeslanou hodnotu do pinového portu a může nastavovat výstupní hodnotu pinů.

Mikrokontrolér PIC24FV16KM202 má dva vstupní/výstupní porty, které jsou označeny PORT A a PORT B. Oba porty jsou 8bitů široké. Značení portů je RXn, kde X je název portu a n je číslo bitu. Např., RA2 je bit 2 portu A, obdobně, RB0 je bit 0 portu B atd. Piny portů jsou obousměrné a vstupní piny lze snadno změnit na výstupní a obráceně. Směr pinů je nutné nakonfigurovat před použitím, pomocí registru speciálních funkcí TRIS. Tento registr nastaví směr jednotlivých pinů portu. Každý port má příslušný TRIS registr. Tzn., TRIS registr PORTU A je TRISA. Pokud nastavíme nízkou hodnotu pinu v TRIS registru dojde k tomu, že pin se stane výstupem. Na obrázku č. 22 je uveden příklad nastavení PORT A, kde piny 1, 2 a 3 jsou výstupy a zbytek pinů jsou vstupy.



Obrázek 22 - Nastavení TRIS registru [7]

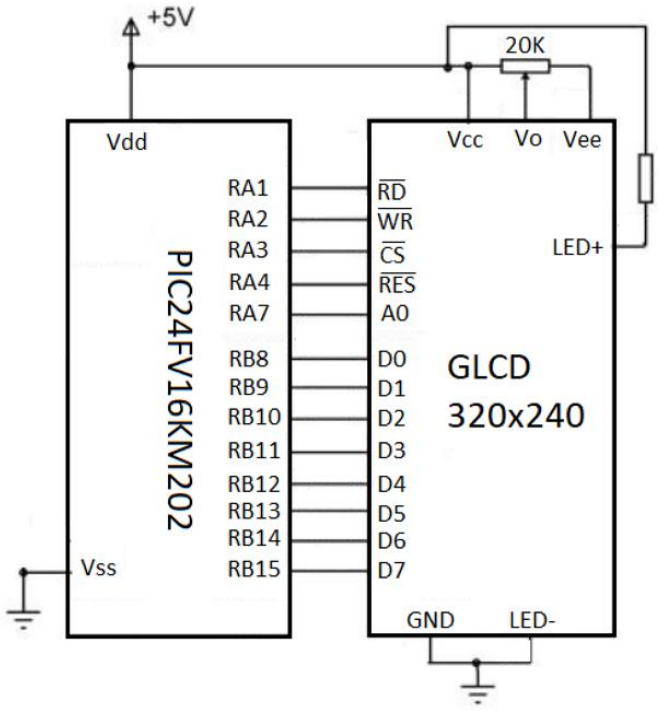
TRISA = 0x7C

3 Praktická část

V této části je představen projekt založený na mikrokontroléru PIC. Bylo navrženo komunikační rozhraní mezi mikrokontrolérem PIC24FV16KM202 a grafickým displejem RG320240B-BIW-V. V kapitole 4.1 jsou popsány vytvořené funkce, které jsou potřeba ke komunikaci s grafickým displejem, jejich popis, co dělají a k čemu slouží. Následně se provedlo ověření funkčnosti mikrokontroléru viz kapitola č. 4.3. Jakmile se ověřila správná funkčnost všech pinů, pokračovalo se v zapojení grafického displeje. Po připojení displeje se začaly testovat jednotlivé funkce, které byly naprogramovány. V práci jsou popsány možné problémy a na závěr kapitoly č. 4 jsou ukázány funkční algoritmy, které vykreslují základní obrazce na displej.

3.1 Schéma zapojení

Na obrázku č. 23 je navržené schéma zapojení grafického displeje a mikrokontroléru.



Obrázek 23 - Schéma zapojení

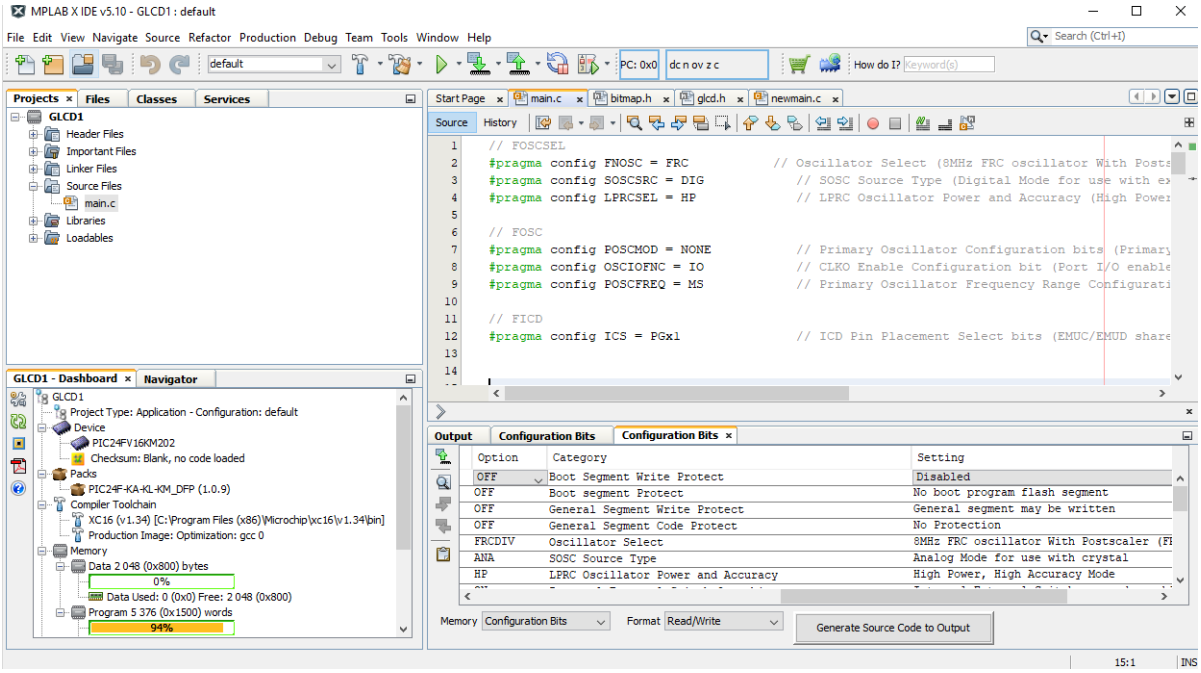
V tabulce č. 4 jsou vypsány piny mikrokontroléru a jejich příslušné piny grafického displeje.

Tabulka 4 - Propojení pinů

Piny mikrokontroléru	Piny grafického displeje
RA1	$\overline{RD}$
RA2	$\overline{WR}$
RA3	$\overline{CS}$
RA4	$\overline{RES}$
RA7	A0
RB8	D0
RB9	D1
RB10	D2
RB11	D3
RB12	D4
RB13	D5
RB14	D6
RB15	D7

3.2 Programování funkcí

K vytvoření programu byl použit software MPLAB X IDE, na obrázku č. 24 je ukázka tohoto prostředí. V levé horní části se nachází struktura programu. Pod tímto oknem je zobrazený název mikrokontroléru, využitý kompilátor a stav paměti mikrokontroléru. V pravém horním okně je zobrazena část programu, kde jsou vygenerované funkce pinů. Pod tímto oknem se nachází generátor pinových funkcí.



Obrázek 24 – software MPLAB X IDE

Následující funkce inicializuje mikrokontrolér, dojde k nastavení směru příslušných pinů a signály se změní na digitální. Navíc dojde k nastavení počátečních hodnot viz funkce glcd_init_graphics (zdrojový kód č. 1).

```
//funkce, která nastaví piny a jejich počáteční hodnoty

void glcd_init_graphics(void)
{
    TRISB = 0;              // port B je nastaven na výstup
    TRISAbits.TRISA3 = 0;   // řídící signál čipu CS - výstup
    TRISAbits.TRISA4 = 0;   // signál pro reset RES - výstup
    TRISAbits.TRISA7 = 0;   // řídící signál A0  - výstup
    TRISAbits.TRISA2 = 0;   // zápis RW - výstup
    TRISAbits.TRISA1 = 0;   // čtení RD - výstup
    TRISAbits.TRISA0 = 0;
    ANSA = 0x00;            //nastavení portu A na digitální signál
    ANSB = 0x00 << 8;       //nastavení portu B na digitální signál
    // Nastavení počátečních hodnot pinů
    LATAbits.LATA3 = 1;     // CS, výchozí hodnota
    LATAbits.LATA4 = 1;     // Reset, výchozí hodnota
    LATB = 0x40 << 8;
    LATAbits.LATA7 = 1;     //A0, výchozí hodnota
    LATAbits.LATA1 = 1;     //RD, výchozí hodnota
}
```

Zdrojový kód 1 - Inicializace displeje

Aby bylo možné odeslat příkaz do ovladače displeje, je zapotřebí následující sekvence signálů, ve kterých dojde k zadání příkazu pomocí vstupní proměnné data viz zdrojový kód č. 2. Tato proměnná může mít např. hodnotu 0x40, což je příkaz pro nastavení pozice kurzoru v displeji.

```
//funkce sloužící k odeslání příkazu, který je zadán pomocí proměnné data
void glcd_cmdwrite(unsigned char data)
{
    A0 = 1;                 //nastavení řídícího signálu na vysokou hodnotu
    lcd_DATA = data << 8;   //data jsou poslaná na port a posunutý
    CS = 0;                 //řídící signál čipu na nízkou hodnotu
    WR = 0;                 //signál pro zápis na nízkou hodnotu
    WR = 1;                 //dojde k odeslání příkazu
    CS = 1;                 //řídícího signálu čipu na vysokou hodnotu
}
```

Zdrojový kód 2 - Funkce příkazu pro zápis

U některých příkazů musí následovat data, která jsou potřebná k provedení příkazu. Např. nastavení souřadnice kurzoru je provedeno dvěma bajty, takže pokud má být adresa kurzoru 0x1B58, nejprve se odešle první bajt 0x58 a poté druhý bajt 0x1B. K tomuto účelu je funkce glcd_datawrite viz zdrojový kód č. 3.

```
// funkce odeslání dat se vstupní proměnnou data
void glcd_datawrite(unsigned char data)
{
    A0 = 0;                 // ukončení předchozí funkce
    lcd_DATA = data << 8;   // nastavení hodnot k zápisu a posunutí
    CS = 0;                 // nastavení řídícího signálu
    WR = 0;                 // nastavení signálu pro zápis
    WR = 1;                 // potvrzení dat
    CS = 1;                 // nastavení řídícího signálu na vysokou hodnotu
    A0 = 1;                 // ukončení funkce
}
```

Zdrojový kód 3 - Odeslání dat

Pomocí funkcí glcd_cmdwrite a glcd_datawrite viz zdrojové kódy č. 2 a 3 lze nastavit parametry displeje jako jsou výška a šířka znaku, počet řádků (závisí na výšce znaku), textové a grafické vrstvy displeje, s tím souvisí i přiřazení paměti atd.

Funkce glcd_dataread ve zdrojovém kódu č. 4 slouží k načtení dat z paměti displeje, kde nejprve dojde k nastavení pinů na výstup, poté jsou odeslány signály ve správném pořadí a data, která odešle displej do PORTB jsou uloženy v dočasně

proměnná data. Příkaz je ukončen, piny jsou zpět přepnuty na výstup a funkce vrátí proměnou data.

```
//vytvoření funkce pro čtení dat, která vrací proměnnou data
unsigned char glcd_dataread(void)
{
  TRISB = 0xFF << 8; //nastavení portu B na vstup
  unsigned int data; //vytvoření proměnné data
  A0 = 1;            //nastavení řídícího signálu na vysokou hodnotu
  CS = 0;            //nastavení řídícího signálu čipu na nízkou hodnotu
  RD = 0;            //nastavení signálu pro čtení na nízkou hodnotu
  PORTB = 0x00 << 8;    //reset hodnot portu B
  data = PORTB >> 8; //hodnoty se uloží do proměnné data z portu B
  RD = 1;            //nastavení signálu pro čtení na vysokou hodnotu
  CS = 1;            //nastavení řídícího signálu čipu na vysokou hodnotu
  A0 = 1;            //nastavení řídícího signálu na vysokou hodnotu
  TRISB = 0x00 << 8; //zpětné nastavení portu B na výstup
  return (data);     //funkce vrátí proměnnou data
}
```

Zdrojový kód 4 - Funkce čtení dat z paměti displeje

Obdobným způsobem byla vytvořena funkce glcd_cmdread viz zdrojový kód č. 5. Tato funkce kontroluje, zdali kontrolér displeje nedokončuje předchozí operaci.

```
unsigned char glcd_cmdread(void)
{
   unsigned int data;      //vytvoření dočasné proměnné
   TRISB=0xFF<<8;          //nastavení pinů na vstup
   A0 = 0;
   CS = 0;
   RD = 0;
   PORTB = 0x00 << 8;    //reset hodnot portu B
   __delay_us(3);          //pauza
   data = PORTB >> 8;      //uložení načtených dat
   RD = 1;
   CS = 1;
   A0 = 0;
   TRISB=0x00<<8;          //nastavení dat na výstup
   return(data);           //funkce vrátí stav kontroléru
}
```

Zdrojový kód 5 - Funkce čtení dat z paměti displeje

Aby bylo možné kontrolovat stav kontroléru, bylo nutné vytvořit cyklus, který spouští funkci glcd_cmdread viz zdrojový kód č. 6. Tato funkce se opakuje do té doby, než se zpřístupní paměť.

```
void checkbusy(void)
{
 unsigned char busy;                    //dočasná proměnná
 do
 {
    busy =(glcd_cmdread() & 0x40);      //zjistí stav kontroléru a provede
                                        //bitový součin
 }while(busy);                    //cyklus se provádí, dokud platí podmínka
}
```

Zdrojový kód 6 - Funkce kontrola stavu kontroléru

Po spuštění displeje jsou spuštěny funkce, které nastaví všechny parametry displeje. Tyto funkce jsou uvedeny s komentáři ve zdrojovém kódu č. 7.

```
void glcd_systemset(void)
{
    glcd_cmdwrite(0x40);  //Nastavení systému
    glcd_datawrite(0x30); // M0: Interní CG ROM
                          // M1: 160 znaků, 5x7 pixelů
                          // M2: výška řádku 8 pixelů
                          // W/S: jeden panel
                          // IV: bez kompenzace prvního řádku
    glcd_datawrite(0x87); // FX: šířka znaku je 8 pixelů
                          // WF:
    glcd_datawrite(0x07); // FY: výška znaku je 8 pixelů
    glcd_datawrite(0x27); // C/R: 39 bajtů na řádek
    glcd_datawrite(0x2B); // TC/R: celkový počet bajtů na řádek = 43
                          // fosc = 8.0 MHz, ffr = 64 Hz
    glcd_datawrite(0xEF); // L/F: 239 počet řádků
    glcd_datawrite(0x28); // AP: Nastavuje se počet virtuálních řádku 40
    glcd_datawrite(0x00);
}


//Nastavení paměti
void glcd_scroll(void)
{
    glcd_cmdwrite(0x44);
    glcd_datawrite(0x00); // Počáteční adresa prvního bloku, celkem 1200 bajtů
    glcd_datawrite(0x00); // 0x0000
    glcd_datawrite(0xF0); // Počet řádku prvního bloku F0=240
    glcd_datawrite(0xB0); // Počáteční adresa druhého bloku celkem 9600 bajtů
    glcd_datawrite(0x04); // Set 0x04B0
```

```
    glcd_datawrite(0xF0); // Počet řádku druhého bloku F0=240
    glcd_datawrite(0x00);
    glcd_datawrite(0x00);
    glcd_datawrite(0x00);
    glcd_datawrite(0x00);
}

//Nastavení offsetu textové vrstvy vůči grafické
void glcd_hdotscroll(void)
{
    glcd_cmdwrite(0x5A);    // Odeslání příkazu
    glcd_datawrite(0x00);   // Ofset o 0 pixelů
}

//Nastavení textového/grafického módu
void glcd_overlay(void)
{
    glcd_cmdwrite(0x5B);    // Odeslání příkazu
    glcd_datawrite(0x00);   // Zobrazení textové nebo grafické vrstvy
}


void glcd_setcursor(void)
{
    glcd_cmdwrite(0x46);    // Nastaveni kurzoru na souřadnice [0,0]
    glcd_datawrite(0x00);   // Počáteční adresa textové vrstvy
    glcd_datawrite(0x00);

    glcd_cmdwrite(0x5D);    // Příkaz pro nastavení typu kurzoru
    glcd_datawrite(0x05);   // Nastaveni kurzoru šířka 5 pixelů
    glcd_datawrite(0x86);   // Blokový kurzor, výška 6 pixelů
}
```

Zdrojový kód 7 – Nastavení parametrů displeje

Pro vymazání grafické vrstvy lze použít funkci glcd_clear_graphics viz zdrojový kód č. 8. Funkce glcd_setpixelxy nastaví kurzor na souřadnice x=0, y=0, což odpovídá adrese 0x04B0 (počátek grafické vrstvy), poté se nastaví směr pohybu kurzoru doprava příkazem 0x4C a následně se začne zapisovat do paměti for cyklem hodnota 0x00, která se zapisuje až do konce grafické vrstvy končící na adrese 0x2A30. Obdobně lze vymazat textovou vrstvu s tím rozdílem, že se liší počáteční a konečná adresa textové vrstvy. Dále se paměť vyplňuje for cyklem hodnotou 0x20 (mezerou).

```
void glcd_clear_graphics(void)
{
    unsigned int i;
    glcd_setpixelxy(0,0);           // počátek grafické vrstvy v paměti
    glcd_cmdwrite(0x4C);            // směr pohybu kurzoru
    glcd_cmdwrite(0x42);            // příkaz zápisu do paměti
    for (i = 0; i < 0x2A30; i++)    // for cyklus až do konce grafické vrstvy
        glcd_datawrite(0x00);       // vynulování
}
```

Zdrojový kód 8 - Vymazání grafické a textové vrstvy

Zobrazení pixelů na displej zajišťuje funkce glcd_setpixel viz zdrojový kód č. 9. Vstupní proměnné x a y jsou přepočteny na příslušnou adresu a uloženy do proměnné data. Následně se zjistí, který pixel se má zobrazit, což je řešeno posunem posledního bitu na požadovanou pozici. Poté se nastaví pozice kurzoru a odečtou se předchozí data pomocí funkce glcd_dataread a uloží se do proměnné olddata. Do nové proměnné newdata se vloží stará a nová data a dojde k uložení do paměti. Výsledkem je zobrazení pixelu na displeji, který zadá uživatel a nedojde k přepsání předchozích pixelů.

```
void glcd_setpixel(unsigned int x, unsigned int y)
{
    unsigned int data = 1200 + y * 40 + x / 8; // přepočet souřadnic na adresu
    unsigned int posun = x % 8;                // o kolik se má pixel posunout
    unsigned int newdata = 0x80 >> posun;      // posunutí pixelu
    glcd_cmdwrite(0x46);                       // příkaz pro nastavení kurzoru
    glcd_datawrite(data);                      // nastavení adresy
    glcd_datawrite(data >> 8);
    glcd_cmdwrite(0x43);                       // nastavení kurzoru pro čtení
    unsigned int olddata = glcd_dataread();    // uložení předchozích dat
    newdata = (newdata) | (olddata);           // sjednocení dat
    glcd_cmdwrite(0x46);                       // příkaz pro nastavení kurzoru
    glcd_datawrite(data);                      // nastavení adresy
    glcd_datawrite(data >> 8);
    glcd_cmdwrite(0x42);                       // příkaz pro zápis do paměti
    glcd_datawrite(newdata);                   // zapíše pixely
}
```

Zdrojový kód 9 - Funkce k zobrazení pixelu

Dále byla naprogramována funkce pro vykreslení úsečky na základě informací z kapitoly č. 2.2.1. Z důvodu dlouhého kódu byla zobrazena pouze část algoritmu viz zdrojový kód č. 11. Vstupem této funkce jsou dva body se souřadnicemi x a y. Funkce nejprve vypočítá směrnici úsečky, a poté se vykreslí první pixel. V závislosti na znaménku predikce se určují následující pixely a pomocí while cyklu se zobrazí zbytek úsečky na displeji. Tato část algoritmu je pouze pro x-ovou řídící osu. Zbylá část algoritmu je v přílohách na CD.

```
// funkce pro vykreslení úsečky se vstupy pro počáteční a koncový bod
void draw_line(int x1, int y1, int x2, int y2) {
    int deltax, deltay, dx, dy, x, y, x_konc, y_konc, predikcex,
predikcey; //vytvoření proměnných
    deltax = abs(x1 - x2);      //výpočet delta x, absolutní hodnota
    deltay = abs(y1 - y2);      //výpočet delta y, absolutní hodnota
    dx = x2 - x1;               //výpočet delta x
    dy = y2 - y1;               //výpočet delta y
    predikcex = 2 * deltay - deltax;    //výpočet predikcí pro obě osy
    predikcey = 2 * deltax - deltay;
    if (deltay <= deltax) {     //řídící osa x
        if (dx >= 0) {          //vykreslování z leva do prava
            x = x1;             //počáteční x-ová souřadnice
            y = y1;             //počáteční y-ová souřadnice
            x_konc = x2;        //koncová x-ová souřadnice
        }
        else {                  //nastavení počátečního bodu
            x = x2;             //počáteční x-ová souřadnice
            y = y2;             //počáteční y-ová souřadnice
            x_konc = x1;        //koncová x-ová souřadnice
        }
        glcd_setpixel(x, y);    //vykreslení prvního pixelu
        while (x < x_konc) {    //cyklus vykreslování
            if (predikcex < 0) {    //podmínka záporné predikce
                predikcex += 2 * deltay;    //výpočet predikce
            }
            else {
                if ((dx < 0 && dy < 0) || (dx > 0 && dy > 0)) {
                    y++;            //přírůstek v ose y o 1
                }
                else {
                    y--;            //snížení souřadnice y o 1
                }
                predikcex += 2 * (deltay - deltax); //výpočet predikce
            }
            x++;                    //přírůstek v ose x o 1
            glcd_setpixel(x, y);    //vykreslí pixel v daném cyklu
        }
    }
    else {                  //řídící osa y
```

Zdrojový kód 10 – Funkce pro vykreslení úsečky

Jako druhá funkce byl vytvořen algoritmus pro tvorbu kružnice na základě informací z kapitoly č. 2.2.2. Ukázka tohoto algoritmu je ve zdrojovém kódu č. 12. Vstupem této funkce jsou centrální bod a poloměr kružnice. Poté je určen výchozí bod, ze kterého jsou odvozeny symetrické body a počáteční predikce. Dále dle znaménka predikce se sníží souřadnice y o jedničku nebo dojde k navýšení x-ové souřadnice o jedničku a vypočte se následující predikce. Ze souřadnic se poté vykreslí 8 pixelů na displej a cyklus se opakuje do té doby, než se vykreslí celá kružnice.

```
//Funkce pro vykreslení kružnice, která je definována středovým bodem
a poloměrem
void draw_circle(unsigned int xc,unsigned int yc,unsigned int r) {
    int  x, y, dx, dy, predikce;        //vytvoření proměnných
    x = 0;                              //výchozí souřadnice x
    y = r;                              //výchozí souřadnice y
    predikce = 1 - r;                   //výpočet počáteční predikce
    dx = 3;
    dy = 2 * r - 2;
    while (x <= y) {                    //cyklus vykreslování
        glcd_setpixel(xc + x,yc + y);   //vykreslení symetrických bodů
        glcd_setpixel(xc - x,yc + y);
        glcd_setpixel(xc + x,yc - y);
        glcd_setpixel(xc - x,yc - y);
        glcd_setpixel(xc + x,yc + y);
        glcd_setpixel(xc - x,yc + y);
        glcd_setpixel(xc + x,yc - y);
        glcd_setpixel(xc - x,yc - y);
        if (predikce >= 0) {            //podmínka kladné predikce
            predikce = predikce - dy;   //výpočet následující predikce
            dy = dy - 2;
            y--;                        //snížení souřadnice y o 1
        }
        else {                          //záporná predikce
            predikce = predikce + dx;   //výpočet následující predikce
            dx = dx + 2;
            x++;                        //přírůstek v ose x o 1
        }
    }
}
```

Zdrojový kód 11 – Funkce pro vykreslení kružnice

Jako poslední byla vytvořena funkce pro generování pixelů elipsy z teoretických poznatků viz kapitola č. 2.2.3. Ukázka tohoto algoritmu je ve zdrojovém kódu č. 13. Vstupem této funkce jsou souřadnice středového bodu a dvě poloosy. Algoritmus se dělí na dvě části. První část má řídící osu x, kde se vykreslí horní a dolní část elipsy. Druhá část má řídící osu y, kde dojde k zobrazení levé a pravé části elipsy. Nejprve dojde k vytvoření proměnných x a y. Následně dojde k vypočtení kvadrátů poloos a potřebných konstant. Poté je vypočtena predikce pro první bod a pomocí while cyklu jsou vykreslovány symetrické body. Dle znaménka predikce je snížena souřadnice y nebo zvýšena souřadnice x. Cyklus se opakuje, dokud nedojde k vykreslení horní a dolní části elipsy. Obdobným způsobem je vykreslena druhá část algoritmu s tím rozdílem, že řídící osa je x-ová.

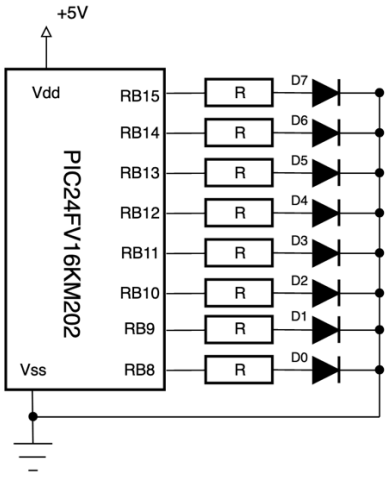
```
//Funkce pro vykreslení elipsy, která je definována středovým bodem a dvěmi
poloosami
void draw_ellipse(unsigned int xc, unsigned int yc, unsigned int a, unsigned b)
{
      int x = 0;                        //počáteční x-ová souřadnice
      int y = b;                        //počáteční y-ová souřadnice
      long int A_kvadrat = a * a;       //výpočet kvadrátu poloosy a
      long int B_kvadrat = b * b;       //výpočet kvadrátu poloosy b
      long int Dve_A_kvadrat = 2 * A_kvadrat;     //výpočet konstanty kvadrátu a
      long int Dve_B_kvadrat = 2 * B_kvadrat;     //výpočet konstanty kvadrátu b
      long int predikce = B_kvadrat - A_kvadrat * b + A_kvadrat / 4;  //výpočet
predikce pro první bod
      long int dx = 0;                          //konstanta dx
      long int dy = Dve_A_kvadrat * b;          //konstanta dy
      while (dx < dy)                       //cyklus vykreslování, řídící osa x
          {
              glcd_setpixel(xc + x, yc + y);  //vykreslení symetrických bodů
              glcd_setpixel(xc - x, yc + y);
              glcd_setpixel(xc + x, yc - y);
              glcd_setpixel(xc - x, yc - y);
              if (predikce >= 0)                //podmínka kladné predikce
              {
                  y--;                          //snížení souřadnice y o 1
                  dy = dy - Dve_A_kvadrat;
                  predikce = predikce - dy;   //výpočet následující predikce
              }
              else                              //záporná predikce
              {
                  x++;                          //přírůstek v ose x o 1
                  dx = dx + Dve_B_kvadrat;
                  predikce = predikce + B_kvadrat + dx;   //výpočet následující
predikce
              }
          }
      //výpočet predikce pro první bod
      predikce = predikce + (3 * (A_kvadrat - B_kvadrat)/2-(dx + dy )) / 2;
      while (y >= 0)                        //cyklus vykreslování, řídící osa y
          {
              glcd_setpixel(xc + x, yc + y);  //vykreslení symetrických bodů
              glcd_setpixel(xc - x, yc + y);
              glcd_setpixel(xc + x, yc - y);
              glcd_setpixel(xc - x, yc - y);
```

```
            if (predikce <= 0)              //podmínka záporné predikce
            {
                x++;                        //přírůstek v ose x o 1
                dx = dx + Dve_B_kvadrat;
                predikce = predikce + dx;   //výpočet následující predikce
            }
            else                            //kladná predikce
            {
                y--;                        //snížení souřadnice y o 1
                dy = dy - Dve_A_kvadrat;
                predikce = predikce + A_kvadrat - dy;   //výpočet následující
predikce
            }
        }
    }
```

Zdrojový kód 12 – Funkce pro vykreslení kružnice

3.3 Ověření funkčnosti mikrokontroléru

Funkčnost mikrokontroléru byla ověřena pomocí 8 LED diod podle schéma zapojení na obrázku č. 25, které simuluje 8bitovou sběrnici. Jako předřadný odpor u všech diod byl zvolen odpor s hodnotou 150Ω.



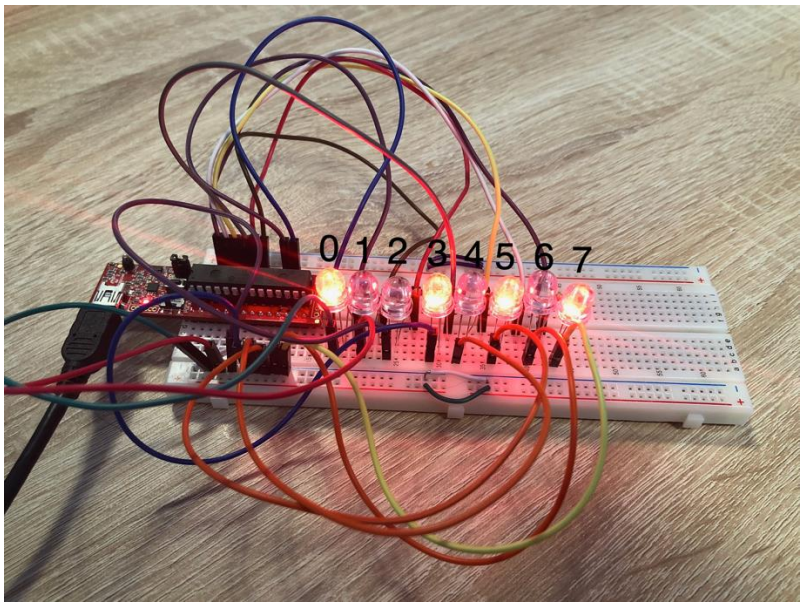
Obrázek 25 - Schéma simulace datové sběrnice

Pomocí jednoduchého kódu viz zdrojový kód č. 8 lze rozsvěcet a zhasínat příslušné diody. Program tvoří nekonečný while cyklus, kde dochází k nastavování registru LATB s prodlevou 200ms. Nejprve dojde k odeslání hodnoty 0xA9 do registru LATB, což odpovídá hodnotě 0b10101001. Poté program počká 200ms, aby bylo možné pozorovat změnu a následně se zhasnou rozsvícené diody resetováním registru LATB. Dále dojde znovu k pauze a program se opakuje.

```
while(1)                    // nekonečný while cyklus
    {
       LATB = 0xA9 << 8;    // nastavení LATB hodnotou 0xA posunutou o 8 bitů
       __delay_ms(200);     // pauza
       LATB = 0x00 << 8;    // resetování LATB
       __delay_ms(200);     // pauza
    }
```

Zdrojový kód 13 - Nekonečný while cyklus

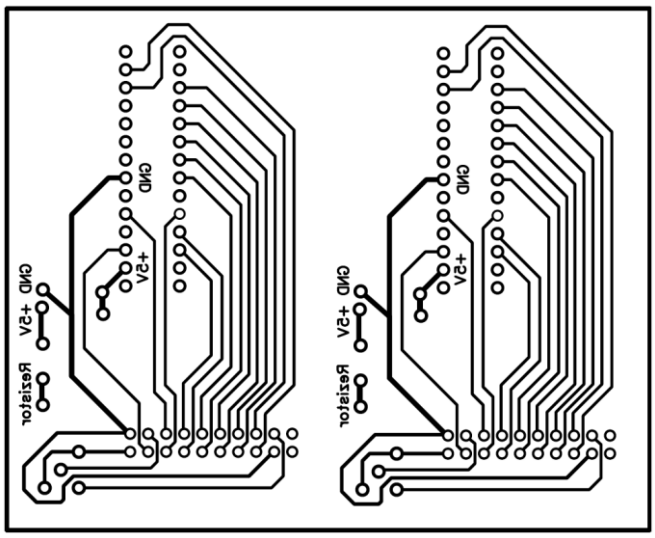
Na obrázku č. 26 lze vidět funkční ukázku zapojení dle schématu na obrázku č. 24. Výsledkem je rychlé blikání diod s indexem 0, 3, 5 a 7 (0b10101001), zbylé diody jsou pořád zhasnuté. Aby bylo možné diody vyfotografovat, bylo nutné program pozastavit.



Obrázek 26 - Simulace datové sběrnice

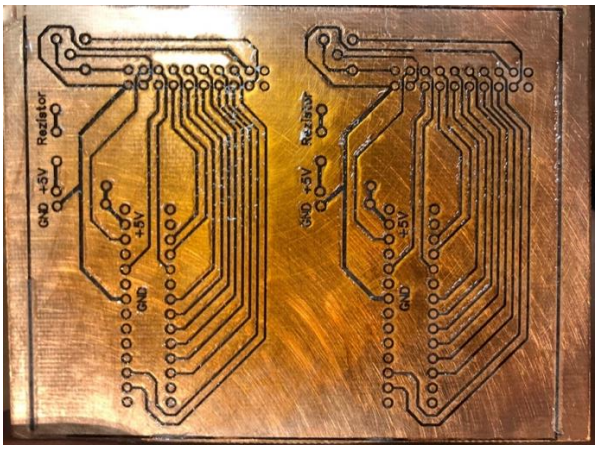
3.4 Návrh zapojení displeje s mikrokontrolérem

Návrh schématu byl vytvořen v online editoru EasyEDA a následně byl vyexportován obrázek ve formátu pdf viz obrázek č. 27. Dále bylo schéma vytištěno laserovou tiskárnou HP laserJet M15w na lesklý křídový papír s měřítkem 1:1.



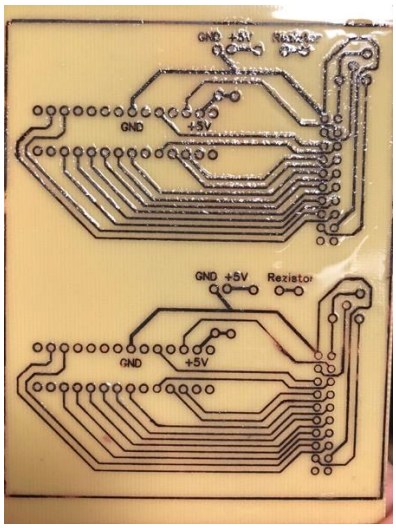
Obrázek 27 - Propojení mikrokontroléru s IDC konektorem

Schéma bylo naženleno na měděnou destičku (obrázek č.28) a opatrně očištěno od zbytků papíru, aby se zaschlý toner nepoškodil. Případně lze provést korekce lihovým fixem.

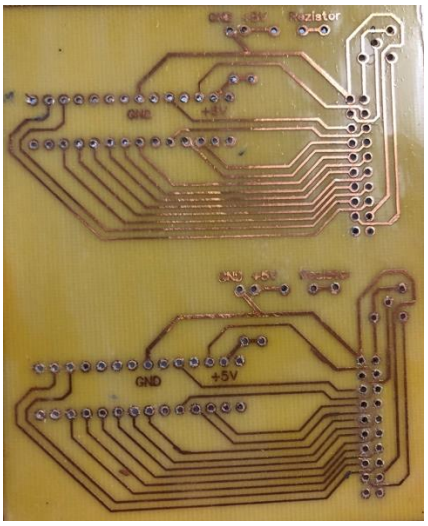


Obrázek 28 - Naženlené schéma na měděnou destičku

V dalším kroku bylo provedeno odleptání přebytečné měděné vrstvy v leptací lázni persíranu sodného. Výsledek leptání je zachycen na obrázeku č. 29. Následně byl očištěn toner brusným papírem a byly vyvrtány dírky pomocí mikrovrtačky viz obrázek č. 30.

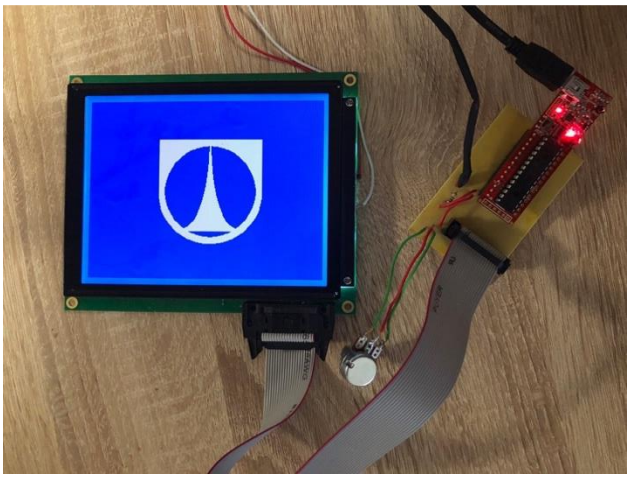


Obrázek 29 – Výsledek po leptání



Obrázek 30 – Výsledný plošný spoj

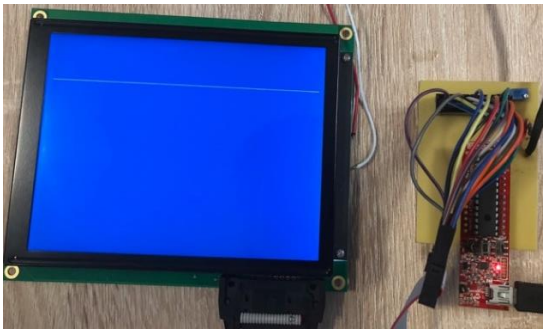
K zapojení displeje bylo zapotřebí plochého kabelu s IDC konektory a IDC zásuvky, kterými jsou osazeny plošný spoj a displej. Navíc bylo nutné zapojit LED podsvícení displeje, kde červený drátek je +5V a bílý drátek je GND. Na obrázku č. 31 je ukázka výsledného zapojení a pro ověření funkčnosti byla zobrazena bitmapa loga Technické univerzity v Liberci.



Obrázek 31 – Výsledné zapojení

3.5 Testování

Nejprve byly všechny kontakty na plošném spoji otestovány pomocí multimetru, zda jsou všechny zkoušené kontakty vodivě propojené. Jakmile byla ověřena funkčnost plošného spoje, byl připojen plochý kabel do IDC zásuvky, mikrokontrolér a podsvícení displeje propojeno se zdrojem napětí. Po zapnutí displeje se začala zobrazovat horizontální čára viz obrázek č. 32, která byla způsobena tím, že výchozí hodnota pinu pro RESET byla vysoká. Po odpojení RESET signálu se začala zobrazovat grafická vrstva s maximálním kontrastem viz obrázek č. 33.



Obrázek 32 – Znázorněná chyba resetování

Obrázek 33 – Výsledné zapojení

Dále bylo pomocí funkcí glcd_cmdwrite a glcd_datawrite vykreslena na displej věta Hello world viz obrázek č. 34. Zpočátku se jevilo, že je vše v pořádku, ale když došlo ke zvýšení kontrastu pomocí potenciometru, začal se projevovat tzv. sněžný efekt

viz obrázek č. 35. Tento jev se dá výrazně snížit pomocí funkce busycheck, která kontroluje, zdali ovladač displeje zpracovává úlohu a pokud ne, umožní mikrokontroléru zápis dat do paměti displeje. U tohoto jevu může docházet k zobrazení písmen na náhodné poloze, případně dojde k zobrazení špatných informací.

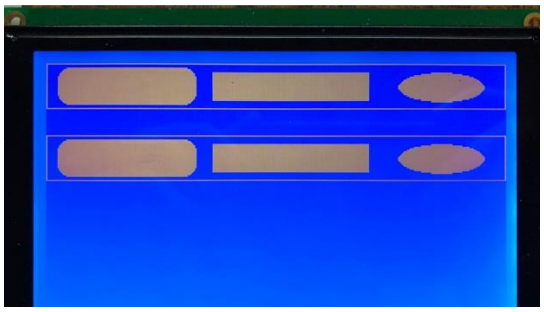
Obrázek 34 – Zobrazení textu

Obrázek 35 – Sněžný efekt

Na obrázku č. 36 je ukázka sněžného efektu po aplikaci funkce busycheck, kde je vidět výrazné zlepšení.

Obrázek 36 – Snížený sněžný efekt

V následující části byla otestována funkce glcd_dataread. Testovací obrázek byl zapsán na souřadnice (0,0) a byl pomocí for cyklu načítán, data byla vždy uložena do dočasné proměnné a následně zapsána s posunutím v ose y do paměti. Výsledek je na obrázku č. 37. Tímto byla ověřena stabilita čtení a mohlo se začít vykreslovat na displej.



Obrázek 37 – Kopírovaní dat

3.6 Vykreslování na displej

Pomocí algoritmu, který byl vytvořen na základě informací z kapitoly č. 3.2.1, byla zobrazena úhlopříčka s počátečním bodem (0,0) a koncovým bodem (319, 239) a horizontální čára na displej viz obrázky č. 38 a 39.

Obrázek 38 – Zobrazení úhlopříčky

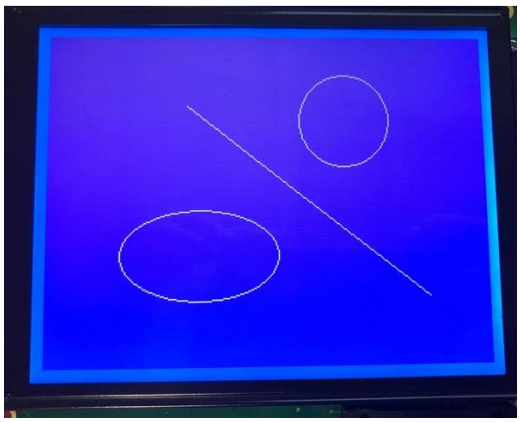
Obrázek 39 – Zobrazení horizontální čáry

Dále byla vytvořena ukázka vykreslení kružnice (obrázek č. 40) pomocí Bresenhamova algoritmu pro tvorbu kružnice viz kapitola č. 3.2.2. Na obrázku č. 41 je zobrazena elipsa, která byla vykreslena na základě informací z kapitoly č. 3.2.3.

Obrázek 40 – Zobrazení kružnice

Obrázek 41 – Zobrazení elipsy

Na obrázku č. 42 jsou zobrazeny současně kružnice, úsečka a elipsa.



Obrázek 42 – Zobrazení elipsy, přímky a kružnice

4 Závěr

Cílem diplomové práce bylo vytvoření grafické knihovny funkcí v programovacím jazyce C. Byly vytvořeny algoritmy pro tvorbu úsečky, kružnice a elipsy. Na základě teoretických znalostí algoritmů a zkušeností z programovacího jazyka C byly vytvořeny funkce pro tvorbu úsečky, kružnice a elipsy. Funkce k vytvoření libovolné úsečky byla naprogramována a část algoritmu pro směrnici m od 0 do 1 byla zobrazena ve zdrojovém kódu č. 10. Zbytek algoritmu se nachází v přílohách na CD. Výsledkem bylo vykreslení úsečky, která byla definována dvěma body. Na obrázku č. 38 je zobrazeno správné vykreslení úsečky. Dále byl vytvořen algoritmus pro tvorbu kružnice. Tento algoritmus je zobrazen ve zdrojovém kódu č. 11 a pomocí tohoto algoritmu lze umístit a vykreslit kružnici na displej, která je definována středovým bodem a poloměrem kružnice. Na obrázku č. 40 je ukázka kružnice, která byla vytvořena pomocí zmíněného algoritmu. Poté byla vytvořena funkce viz zdrojový kód č. 12, kterou lze vykreslit elipsu definovanou centrálním bodem a dvěma poloosami. Příklad vykreslení na displeji je na obrázku č. 41. Vícenásobné zobrazení je na obrázku č. 42, kde jsou dohromady vidět úsečka, kružnice a elipsa. Na obrázku č. 34 byl zobrazen text, což je pouze zápis do textové vrstvy paměti.

Hlavní součástky obvodu tvořily mikrokontrolér PIC24FV16KM202 firmy Microchip a displej RG320240B-BIW-V firmy RAYSTAR. Bylo nutné navrhnout a vytvořit schéma zapojení mezi mikrokontrolérem a displejem, tato část byla velmi obtížná, jelikož z důvodu nekvalitního nepájivého kontaktního pole docházelo k chybám ve vykreslování na displej. Z tohoto důvodu byl vytvořen plošný spoj, aby vzdálenost kabelů mezi displejem a mikrokontrolérem byla co nejkratší. Ukázka výsledného zapojení je na obrázku č. 31.

Návrhem na zlepšení by mohlo být navýšení externí paměti, aby displej mohl zobrazovat více bitmap, jelikož do paměti mikrokontroléru lze uložit pouze jednu bitmapu. Dalším návrhem by byla možnost připojení senzorů k mikrokontroléru a vykreslování zaznamenaných dat na displej. Pokud by byla součást zapojení externí paměť, mohly by se zapisovat data poskytnutými senzory do externí paměti.

Seznam použité literatury

[1] ŽÁRA, Jiří. *Počítačová grafika: principy a algoritmy*. Praha: Grada, 1992. ISBN 80-856-2300-5.

[2] ŽÁRA, Jiří, Bedřich BENEŠ a Petr FELKEL. Moderní počítačová grafika. Praha: Computer Press, 1998. ISBN 80-722-6049-9.

[3] The Bresenham's Line Drawing Algorithm. In: Saloni Baweja [online]. September 14, 2014 [cit. 2019-03-28]. Dostupné z: https://salonibaweja10.wordpress.com/tag/derivation-of-bresenham-line-algorithm/

[4] DION, Francois. Pi-A-Sketch code review. Raspberry Pi Python Adventures [online]. January 16, 2013 [cit. 2019-03-28]. Dostupné z: http://raspberry-python.blogspot.com/2013/01/pi-sketch-code-review.html

[5] KUMAR, Niteesh. Mid point ellipse drawing algorithm in c++. The GEEK mode [online]. [cit. 2019-03-28]. Dostupné z: http://www.thegeekmode.org/2017/02/mid-point-ellipse-drawing.html

[6] FITZHENRY, Sorcha. Circles, Ellipses, Curves: PowerPoint PPT Presentation. In: Slide Serve [online]. Nov 14, 2013 [cit. 2019-03-28]. Dostupné z: https://www.slideserve.com/sorcha/circles-ellipses-curves

[7] IBRAHIM, Dogan. Using LEDs, LCDs, and GLCDs in microcontroller projects. Chichester, West Sussex, U.K.: Wiley, 2012. ISBN 978-111-9940-708.

[8] Diodo Led 5mm. In: Tesla electronics [online]. [cit. 2019-03-28]. Dostupné z: http://www.teslaelectronics.cl/inicio/136-diodo-led-5mm.html

[9] Adafruit LED Backpacks: 0.56" 7-Segment Backpack. In: Adafruit [online]. NYC [cit. 2019-03-28]. Dostupné z: https://learn.adafruit.com/assets/150

[10] Princip. In: HPM wiki: Hardware pro multimédia [online]. Praha, 18-Dec-2013 [cit. 2019-03-28]. Dostupné z: http://noel.feld.cvut.cz/vyu/a2b31hpm/images/a/a0/Princip.jpg

[11] In: Farnell [online]. Premier Farnell UK Limited, 2019 [cit. 2019-03-29]. Dostupné z: https://cz.farnell.com/productimages/large/en_GB/2769717-40.jpg

[12] LCD display / graphic / 16-segment / 8-digit. In: Direct Industry [online]. 2019 [cit. 2019-03-29]. Dostupné z: http://www.directindustry.com/prod/hebei-jiya-electronics-co-ltd/product-120521-1351511.html

[13] LCD Display 16x2, LCD Module 16x2 [online]. In: Taiwan: WINSTAR Display Co., 2018 [cit. 2019-03-29]. Dostupné z: https://www.winstar.com.tw/products/character-lcd-display-module/lcd-display-16x2.html

[14] Graphic LCD Display 320x240, Display LCD 320x240. In: Raystar [online]. Raystar Optronics, 2018 [cit. 2019-03-29]. Dostupné z: https://www.raystar-optronics.com/graphic-lcd-display-module/lcd-display-320x240.html

[15] CROSS, Jason. Digital Displays Explained. In: PCWorld [online]. IDG Communications, March 18, 2012 [cit. 2019-03-29]. Dostupné z: https://www.pcworld.com/article/251988/tablets/digital-displays-explained.html?page=3

[16] KOUTNÝ, Jaroslav a Ivo VLK. Elektronika I učebnice. VYTVOŘENO V RÁMCI PROJEKTU: DIGITÁLNÍ ŠKOLA: ICT VE VÝUCE TECHNICKÝCH PŘEDMĚTŮ, REG. Č. CZ.1.07/1.1.04/01.0137, Vyšší odborná škola a Střední průmyslová škola elektrotechnická, Olomouc 2009

[17] General Purpose, 16-Bit Flash Microcontrollers with XLP Technology Data Sheet. In: Microchip [online]. Microchip Technology, 2013 [cit. 2019-03-29]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/30003030b.pdf

[18] RG320240B-BIW-V. In: Raystar [online]. Taiwan: Raystar Optronics [cit. 2019-03-29]. Dostupné z: https://www.tme.eu/cz/Document/857c24a567133be9bf0b92b9a8926454/RG320240B-BIW-V.pdf

[19] LCD Controller Specification. In: Pacific Display [online]. RAiO Technology, 2008, June 06, 2008 [cit. 2019-03-29]. Dostupné z: https://www.pacificdisplay.com/ics_app%20notes/raio/RA8835_DS_v22.pdf

[20] Microchip Technology PIC24FJ 16-bit Microcontrollers. In: Mouser Electronics[online]. Mouser Electronics, 2019 [cit. 2019-03-31]. Dostupné z: https://cz.mouser.com/new/microchip/microchip-pic24fj/

Seznam příloh

Příloha 1 – Konfigurace pinů rodiny PIC24FVXXKMX02 [17]

Pin	Pin Features	
	PIC24FXXKMX02	PIC24FVXXKMX02
1	MCLR/VPP/RA5	
2	CVREF+/VREF+/DAC1REF+/AN0/C3INC/CN2/RA0	
3	CVREF-/VREF-/AN1/CN3/RA1	CVREF-/VREF-/AN1/RA1
4	PGED1/AN2/CTCMP/ULPWU/C1IND/C2INB/C3IND/U2TX/CN4/RB0	
5	PGEC1/OA1INA/OA2INA/AN3/C1INC/C2INA/U2RX/CTED12/CN5/RB1	
6	OA1INB/OA2INB/AN4/C1INB/C2IND/SDA2/U1RX/TCKIB/CTED13/CN6/RB2	
7	OA1OUT/AN5/C1INA/C2INC/SCL2/CN7/RB3	
8	VSS	
9	OSCI/CLKI/AN13/CN30/RA2	
10	OSCO/CLKO/AN14/CN29/RA3	
11	SOSCI/AN15/U2RTS/U2BCLK/CN1/RB4	
12	SOSCO/SCLKI/AN16/PWRLCLK/U2CTS/CN0/RA4	
13	VDD	
14	PGED3/AN17/ASDA1/SCK2/IC4/OC1E/CLCINA/CN27/RB5	
15	PGEC3/AN18/ASCL1/SDO2/IC5/OC1F/CLCINB/CN24/RB6	
16	AN19/U1TX/INT0/CN23/RB7	AN19/U1TX/C2OUT/OC1A/INT0/CN23/RB7
17	AN20/SCL1/U1CTS/C3OUT/OC1B/CTED10/CN22/RB8	
18	AN21/SDA1/T1CK/U1RTS/U1BCLK/IC2/OC4/CLC1O/CTED4/CN21/RB9	
19	SDI2/IC1/OC5/CLC2O/CTED3/CN9/RA7	
20	C2OUT/OC1A/CTED1/INT2/CN8/RA6	VCAP OR VDDCORE
21	PGED2/SDI1/OC3A/OC1C/CTED11/CN16/RB10	
22	PGEC2/SCK1/OC2A/CTED9/CN15/RB11	
23	DAC1OUT/AN12/HLVDIN/SS2/IC3/OC2B/CTED2/CN14/RB12	DAC1OUT/AN12/HLVDIN/SS2/IC3/OC2B/CTED2/INT2/CN14/RB12
24	OA1INC/OA2INC/AN11/SDO1/OCFB/OC3B/OC1D/CTPLS/CN13/RB13	
25	DAC2OUT/CVREF/OA1IND/OA2IND/AN10/C3INB/RTCC/C1OUT/OCFA/CTED5/INT1/CN12/RB14	
26	DAC2REF+/OA2OUT/AN9/C3INA/REFO/SS1/TCKIA/CTED6/CN11/RB15	
27	VSS/AVSS	
28	VDD/AVDD	

Legend: Values in red indicate pin function differences between PIC24F(V)XXKM202 and PIC24F(V)XXKM102 devices.

Příloha 2 – Specifikace PIC24FVXXKMX02 [17]

Features	PIC24FV16KM204	PIC24FV08KM204	PIC24FV16KM202	PIC24FV08KM202
Operating Frequency	DC-32 MHz			
Program Memory (bytes)	16K	8K	16K	8K
Program Memory (instructions)	5632	2816	5632	2816
Data Memory (bytes)	2048			
Data EEPROM Memory (bytes)	512			
Interrupt Sources (soft vectors/NMI traps)	40 (36/4)			
Voltage Range	2.0-5.5V			
I/O Ports	PORTA<11:7,5:0> PORTB<15:0> PORTC<9:0>		PORTA<7,5:0> PORTB<15:0>	
Total I/O Pins	37		23	
Timers	11 (One 16-bit timer, five MCCPs/SCCPs with up to two 16/32 timers each)			
Capture/Compare/PWM modules MCCP SCCP	 3 2			
Serial Communications MSSP UART	 2 2			
Input Change Notification Interrupt	36		22	
12-Bit Analog-to-Digital Module (input channels)	22		19	
Analog Comparators	3			
8-Bit Digital-to-Analog Converters	2			
Operational Amplifiers	2			
Charge Time Measurement Unit (CTMU)	Yes			
Real-Time Clock and Calendar (RTCC)	Yes			
Configurable Logic Cell (CLC)	2			
Resets (and delays)	POR, BOR, RESET Instruction, $\overline{\text{MCLR}}$, WDT, Illegal Opcode, REPEAT Instruction, Hardware Traps, Configuration Word Mismatch (PWRT, OST, PLL Lock)			
Instruction Set	76 Base Instructions, Multiple Addressing Mode Variations			
Packages	44-Pin QFN/TQFP, 48-Pin UQFN		28-Pin SPDIP/SSOP/SOIC/QFN	

Příloha 3 – Příkazové sety [19]

Class	Command	Code											Hex	Command Description	Command Read Parameters	
		RD	WR	A0	D7	D6	D5	D4	D3	D2	D1	D0			No. of Bytes	Section
System Control	**SYSTEM SET**	1	0	1	0	1	0	0	0	0	0	0	40	Initialize device and display	8	9-2-1
	SLEEP IN	1	0	1	0	1	0	1	0	0	1	1	53	Enter standby mode	0	9-2-2
Display Control	**DISPLAY ON/OFF**	1	0	1	0	1	0	1	1	0	0	D	58, 59	Enable and disable display and display flashing	1	9-3-1
	SCROLL	1	0	1	0	1	0	0	0	1	0	0	44	Set display start address and display regions	10	9-3-2
	CSRFORM	1	0	1	0	1	0	1	1	1	0	1	5D	Set cursor type	2	9-3-3
	CGRAM ADR	1	0	1	0	1	0	1	1	1	0	0	5C	Set start address of character generator RAM	2	9-3-6
	CSRDIR	1	0	1	0	1	0	0	1	1	CD 1	CD 0	4C to 4F	Set direction of cursor movement	0	9-3-4
	HDOT SCR	1	0	1	0	1	0	1	1	0	1	0	5A	Set horizontal scroll position	1	9-3-7
	OVLAY	1	0	1	0	1	0	1	1	0	1	1	5B	Set display overlay format	1	9-3-5
Drawing Control	**CSRW**	1	0	1	0	1	0	0	0	1	1	0	46	Set cursor address	2	9-r1
	CSRR	1	0	1	0	1	0	0	0	1	1	1	47	Read cursor address	2	9-4-2
Memory Control	**MWRITE**	1	0	1	0	1	0	0	0	0	1	0	42	Write to display memory	—	9-5-1
	MREAD	1	0	1	0	1	0	0	0	0	1	1	43	Read from display memory	—	9-5-2