ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta elektrotechnická
Katedra elektrotechniky a informačních technologií

BAKALÁŘSKÁ PRÁCE

Řadiče pro malé LCD zobrazovače

Autor práce: **Tomáš Urban**
Vedoucí práce: **Ing. Petr Weissar, Ph.D.**

2023

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš URBAN**
Osobní číslo:	**E19B0254P**
Studijní program:	**B2612 Elektrotechnika a informatika**
Téma práce:	**Řadiče pro malé LCD zobrazovače**
Zadávající katedra:	**Katedra elektroniky a informačních technologií**

Zásady pro vypracování

1. Prozkoumejte řadiče používané pro „malé“ LCD zobrazovací panely, uvažujte barevné i černobílé varianty
2. Porovnejte podporované rozlišení, barevnou hloubku, způsoby komunikace, napájecí napětí a spotřebu, příp. další parametry
3. Vytvořte vzorové příklady použití v jazyce C/C++ pro mikroprocesory ARM CortexM (příp. ESP32)

Rozsah bakalářské práce: **30 – 40**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **elektronická**

Seznam doporučené literatury:

1. Yiu, Joseph. The Definitive Guide to ARM ®Cortex ®-M3 and Cortex®-M4 Processors, Elsevier Science & Technology, 2013.
2. Online dokumentace výrobců.

Vedoucí bakalářské práce: **Ing. Petr Weissar, Ph.D.**
Katedra elektroniky a informačních technologií

Datum zadání bakalářské práce: **7. října 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

L.S.

Prof. Ing. Zdeněk Peroutka, Ph.D.
děkan

Doc. Ing. Jiří Hammerbauer, Ph.D.
vedoucí katedry

V Plzni dne 7. října 2022

Abstrakt

Bakalářská práce se zabývá rozborem řadičů určených pro LCD zobrazovače. Konkrétně se zaměřuje na řadiče KS0108, ST7565, ST7066, HD44780, ST7735, IL9163, ST7789 a GC9A01. Důraz je kladen na vlastnosti jednotlivých řadičů, které jsou následně porovnány. Ke každému řadiči jsou vytvořené ukázkové příklady použití. Součástí bakalářské práce je i knihovna pro použití na mikroprocesoru STM32F411RE, včetně návodu na zprovoznění knihovny.

Klíčová slova

STM32, LCD řadiče, knihovna pro LCD řadiče

Abstract

The bachelor work analyses controllers designed for LCD displays. Specifically, it focuses on controllers KS0108, ST7565, ST7066, HD44780, ST7735, IL9163, ST7789 and GC9A01. The focus is on the characteristics of each controller, which are then compared. For each controller, demonstration examples of use are created. A library for use on the STM32F411RE microprocessor is also included in the bachelor's work, including instructions on how to make the library work.

Keywords

STM32, LCD drivers, library for LCD drivers

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci, zpracovanou na závěr studia na Fakulty elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem svou závěrečnou práci 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 závěrečné 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 důsledku vyplývajících z ustanovení § 270 trestného zákona č. 40/2009 Sb.

Také prohlasuji, že veškerý software, použitý při řešení teto bakalářské práce, je legální.

V Plzni dne 25. května 2023

Tomáš Urban

. . . . . .

Podpis

Obsah

Seznam symbolů

Veličiny

U	V	elektrické napětí
I	A	elektrický proud
P	W	výkon
f	Hz	kmitočet
t	s	čas

Seznam zkratek

LCD	Liquid Crystal Display
MPU	Microprocessor Unit
RAM	Random Access Memory
SRAM	Static Random Access Memory
ROM	Read-Only Memory
LDO	Low-Dropout Regulator
EPROM	Erasable Programmable Read-Only Memory
DMA	Direct Memory Access
USART	Universal Synchronous / Asynchronous Receiver and Transmitter
UART	Universal Asynchronous Receiver - Transmitter
CAN BUS	Controller Area Network
SPI	Serial Peripheral Interface
SCLK	Clock Signal
CS/SS	Chip Select
MISO	Master In Slave Out
MOSI	Master Out Slave In
QSPI	Queued Serial Peripheral Interface
IIC	Inter-Integrated Circuit
RTC	Real-time clock

1 Úvod do problematiky

Bakalářská práce se zabývá problematikou LCD řadičů pro zobrazovače. Cílem je prozkoumat řadiče používané pro malé LCD zobrazovací panely jak v monochromní tak i v barevné variantě. Následně porovnat vlastnosti jednotlivých řadičů, kde se zaměřit na barevnou hloubku, napájecí napětí, způsoby komunikace a spotřebu. Posledním cílem mé práce je naprogramovat knihovny pro práci s různými řadiči. Knihovny budou psány v programovacím jazyce C s použitím mikroprocesoru ARM Cortex M. K jednotlivým řadičů vytvořit příklady použití.

Na trhu je k dispozici velké množství různých řadičů využívající různé způsoby komunikace. Nejpoužívanějším způsobem komunikace u řadičů pro malé zobrazovací panely je sériová sběrnice SPI případně 8 bitová paralelní sběrnice. U řadičů se obvykle využívají standardní sběrnice avšak jednotlivé řadiče se liší používanými příkazy. Řadiče totiž vyrábějí různí výrobci s různými funkcemi.

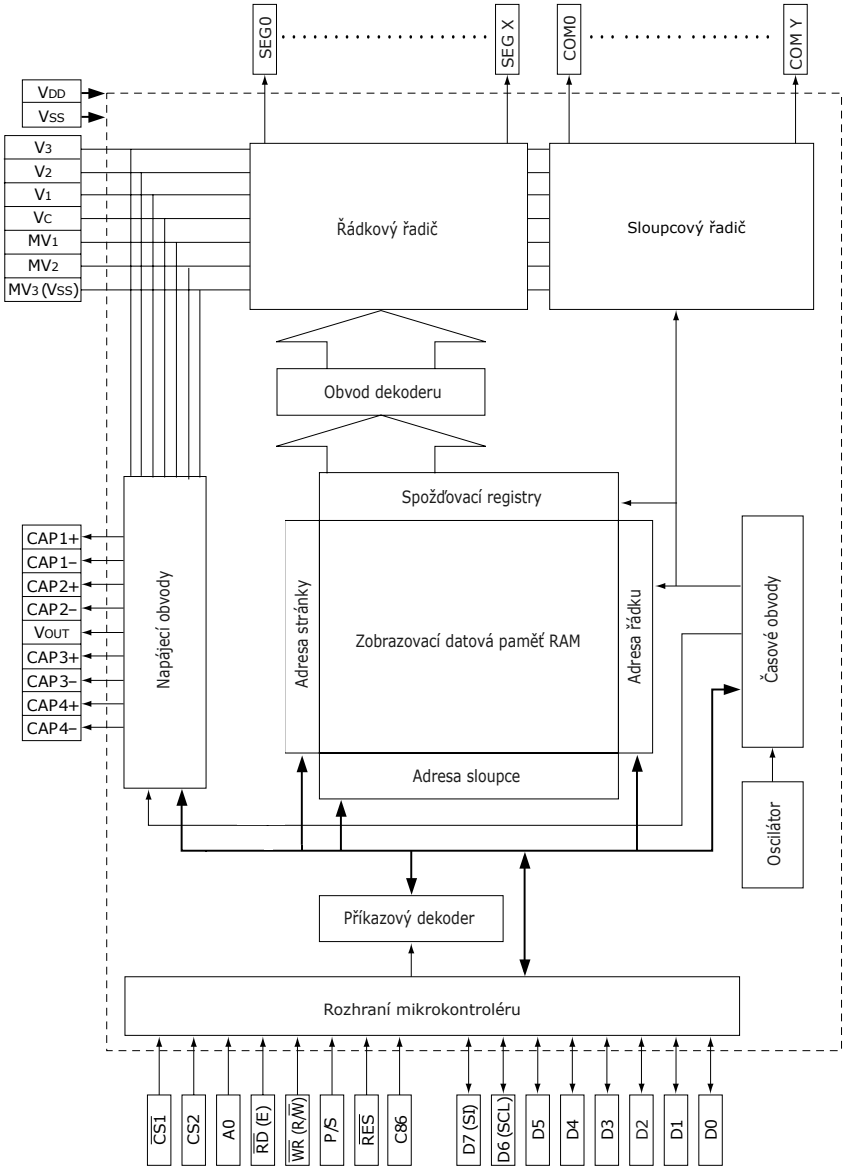
Plánem mé práce je naprogramovat a navrhnout knihovnu, která bude umožňovat jednoduchou práci s LCD řadiči. K realizaci využiji dostupný mikrokontrolér STM32F411RE. Specifické technické informace o tomto mikrokontroléru lze nalézt zde. [16]. Pro řešení tohoto problému lze využít již existující HAL knihovny popřípadě Low Layer (LL) knihovnu. V těchto existujících knihovnách je, ale náročné pracovat z důvodu nedostatečné technické dokumentace.

Již existují knihovny pro řešení komunikace mezi LCD řadičem a mikrokontrolérem. Takže není žádný problém najít jinou existující knihovnu. Mé řešení je troško odlišné od již existujících, kde v první řadě nebudu využívat žádné HAL knihovny nebo Low Layer knihovnu. Místo toho využiji části kódu, s kterými jsme pracovali na cvičení v rámci předmětu Mikroprocesory a počítače. Také nebudu využívat přímého zápisu dat do řadiče, ale nejdříve si vytvořím v RAM paměti mikrokontroléru pole, které odpovídá stejné velikosti jako je velikost zobrazovací plochy. Do tohoto pole si budu ukládat data a následně je posílat do řadiče jako celý snímek. Výhodou tohoto řešení je možnost využití DMA.

Přínosem mé bakalářské práce je zjednodušit práci s LCD zobrazovače-mi na mikroprocesorech STM32. Vniklou knihovnu lze využít pro výukové účely nebo pro projekty používající LCD zobrazovače.

2 Blokové schéma řadiče

Fungování řadiče lze rozložit do několika bloků, kde každý blok je specializován na konkrétní činnost. Počet těchto bloků není nijak pevně stanoven, záleží jen na firmách vyrábějící řadiče a jejích potřeb co mají tyto obvody umožňovat. Na obrázku 2.1 je zobrazeno blokové schéma řadiče S1D15E06. Vnitřní struktura je reprezentativní ukázkou vnitřní funkce řadiče. Většina řadičů pro černobílé displeje pracují obdobně, u řadičů pro barevné zobrazovače je navíc umístěn blok pro úpravu barvy. Na tomto řadiči popíši funkci jednotlivých bloků.



Obr. 2.1 Blokové schéma řadiče S1D15E06 [8]

2.1 Rozhraní mikrokontroléru

Blok rozhraní mikrokontroléru slouží ke komunikaci řadiče s ostatními mikrokontroléry případně i s jinými řadiči. Blok nezastává jen funkci komunikace, ale především zajišťuje řízení celého řadiče a provádí nastavování jednotlivých bloků. Pro komunikaci s vnějším mikrokontrolérem využívá vyvedené sběrnice, na obrázku 2.1 je konkrétně znázorněna 8 bitová paralelní sběrnice. Můžeme se setkat i s 16 nebo 32 bitovou paralelní sběrnicí, ale jen spíše u řadičů pro velké displeje. Pro ušetření počtu vodičů se používají i sériové sběrnice. Nejčastěji se používá SPI sběrnice. Řada řadičů má k dispozici více sběrnic a lze si vybrat jakou sběrnici použít.

2.2 Příkazový dekodér

Příkazový dekodér slouží k dekódování příkazů, které přichází z mikrokontroléru. Po dekódování příkazu, příkaz směruje přes sběrnici do cílového bloku.

2.3 Zobrazovací datová paměť RAM

Zobrazovaný obraz na LCD displeji je uložen ve statických pamětích RAM (SRAM). Jádrem paměti je matice paměťových buněk, umístěných v křížení řádkových a sloupcových vodičů. K zapamatování informace o daném signálu se využívají klopné obvody. Zapamatovaný bit, bude uložen v klopném obvodu tak dlouho dokud nebude přepsán nebo do vypnutí napájení. Pro výběr bitů z dané adresy se používají řádkové a sloupcové dekodéry. Řádkové dekodéry vybírají vždy jen jeden řádek a sloupcové dekodéry řídí výběrové obvody, kde se jedná o CMOS spínače zapojené jako multiplexery/demultiplexery. Na výběrové obvody následují čtecí a zápisové zesilovače, které jsou nutné pro zesílení signálu z paměťových buněk. Velkou výhodou těchto pamětí je vysoká rychlost čtení nebo zápisu, vybavovací doby pod 10 ns. Nevýhodou zůstává potřeba mít 6 tranzistorů na vytvoření jedné buňky. Velikost paměti SRAM nejčastěji odpovídá velikosti zobrazované ploše, ale není tomu vždy. Někteří výrobci LCD zobrazovačů můžou zvolit menší počet pixelů na obrazovce než je velikost paměti SRAM a tím i část této paměti zůstane nevyužita. [13]

2.4 Generátor synchronizačních impulzů

Pro generování synchronizačních impulzů je umístěn na čipu oscilátor. Nejčastěji se jedná o RC oscilátor s pevnou frekvencí, který generuje sinusový signál. Generovaný signál se přivádí do vedlejšího bloku s časovými obvody. Přiváděný signál z oscilátoru je upraven do obdélníkového signálu a v násobičce je vynásoben na několik stovek kilohertz.

2.5 Zobrazování dat na displeji

Proces zobrazení uložených dat v paměti RAM je řízen hodinovým signálem. Nejprve je vybrán příslušný řádek v RAM paměti a její obsah se přesune přes zpožďovací registry do dekodéru, zde je signál dekódován a pokračuje do řádkového řadiče. Než jsou data přesunuta na výstupní brány řadiče je nejprve nutné převést obsah celého vybraného řádku z digitálního signálu na analogový. Odeslání řádku na výstupní piny je doprovázeno i sloupcovým řadičem, který musí nejdříve vybrat patřičný řádek, aby bylo možné celý řádek zobrazit na displeji.

2.6 Napájecí obvody

Ke správnému fungování řadiče k připojenému LCD zobrazovači je potřeba zajistit správné napájecí napětí. Napájecí obvody zajišťují napájení celého řadiče, včetně připojeného LCD displeje. Proto jsou napájecí obvody vybaveny víceúrovňovými napájecího napětí. Nejčastěji se jedná o LDO stabilizátory napětí. Výpočet ztrátového výkonu na LDO stabilizátorech je vyobrazen v rovnici 2.1.

$$P = (U_{IN} - U_{OUT}) \times I \tag{2.1}$$

3 Sběrnice

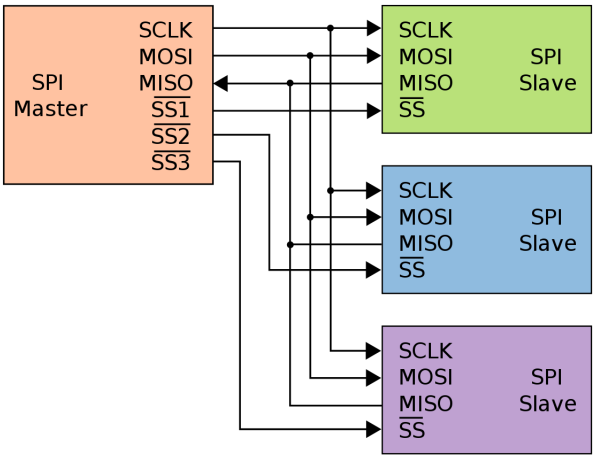
Sběrnice slouží k rychlé a ideálně bezeztrátové komunikaci mezi dvěma periferiemi. Nemusí se jednat jen o řadiče pro LCD zobrazovače, ale existuje velká škála obvodů, s kterými lze komunikovat pomocí sběrnic. Mezi tyto periférie například patří teplotní senzory, gyroskopy, akcelerometry a již zmíněné řadiče pro LCD obrazovky. Mezi nejběžnější sériové komunikační sběrnice patří USART, UART, CAN Bus, SPI, IIC a mezi paralelní sběrnice patří 4/8/12/16/18 bitové paralelní sběrnice. Pro komunikaci mezi řadičem a mikrokontrolérem se nejčastěji používají sériové sběrnice SPI, výjimečně se lze setkat i s IIC sběrnicí. Z paralelních sběrnic nejčastěji 8 bitová sběrnice.

3.1 Sběrnice SPI

Sériové rozhraní SPI patří mezi nejjednodušší se kterými se lze v praxi setkat. Díky své jednoduchosti je velice používaná. Jde o čtyřvodičovou sběrnici, která podporuje Full Duplex komunikaci, tedy zvládá vysílat a přijímat zároveň. Velkou výhodou je velká propustnost, kde hodinové signály mohou dosahovat až 40 MHz, ale i tento limit může být překonán. Díky své flexibilitě není komunikace limitovaná na 8 bitová slova, ale je možné přenášet informace i o větší délce. [10]

Rozhraní nemá jen samé výhody najdou se i nevýhody. Největší nevýhodou je náročnost na počet vodičů při zvyšování se počtu slave zařízení, protože pro každý slave je potřeba přivést CS signál samostatně. Sběrnice SPI zvládne přenášet data jen na krátké vzdálenosti. Nevyužívá žádný protokol pro kontrolu chyb, nelze tedy ověřit zda slave přijal stejnou zprávu jako byla vyslána masterem. Také podporuje jen jednoho mastra na celé sběrnici. [10]

Na obrázku 3.1 je znázorněno propojení jednotlivých prvků přes sběrnici SPI. Master zajišťuje řízení komunikace po sběrnici a jako jediný vysílá po sběrnici hodinový signál podle kterého se řídí časování celé sběrnice. Zahájení komunikace se provádí přivedením logické 0 na vstup CS do slavu. Pokud tomu tak není je vývod MISO ve stavu vysoké vstupní impedance.



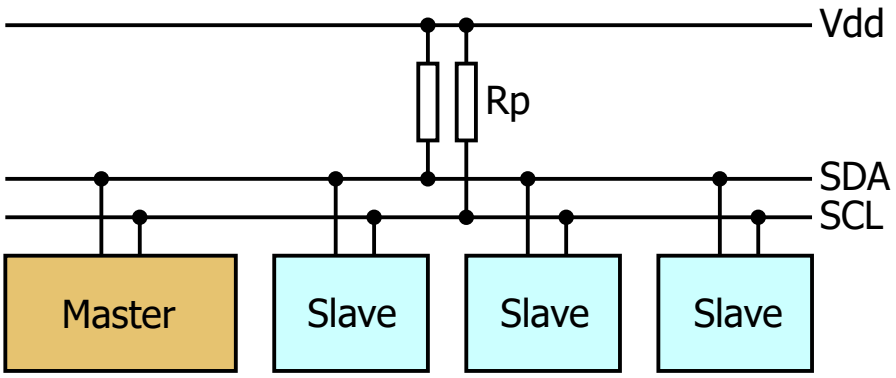
Obr. 3.1 Znázornění zapojení sběrnice SPI [5]

Existují další možné modifikace sběrnice SPI. Jedním s možných modifikací je tzv. Dual SPI, ze sběrnice se stane half-duplex, to znamená, že může vysílat jen jedno zařízení v jeden okamžik. Ovšem na druhou stranu se data posílají po dvou vodičích, takže se za jeden hodinový cyklus přenesou 2 bity. Další možnou modifikací je takzvané QSPI to obnáší rozšíření sběrnice o další dva vodiče. Díky tomu lze posílat více dat za jeden cyklus hodinového impulzu. Sběrnici lze nakonfigurovat, že bude pracovat v režimu full-duplex nebo half-duplex. To dovoluje v režimu half-duplex přenést až 4 bity za jeden hodinový cyklus. [10]

3.2 Sběrnice IIC

Sériová sběrnice IIC byla vyvinutá společností Philips. Sběrnice našla uplatnění v nejrůznějších senzorech a periferiích, které nevyžadují přenášet velké datové objemy. Do této kategorie patří například teplotní senzory, sérioparalerní převodníky, hodiny reálného času (RTC) a mnoho dalších periférií.

Velkou výhodou sběrnice je nenáročnost na počet vodičů, pro zprovoznění sběrnice stačí propojit všechny členy jen dvěma vodiči, na sběrnici lze připojit až 128 zařízení. Sběrnice může komunikovat jen s jedním zařízením v jeden okamžik, tzv. half-duplex. Propojení jednotlivých členů na sběrnici IIC je znázorněno na obrázku 3.2. V tomto případě je zde umístění jeden master, ovšem na sběrnici lze umístit více mastrů. [9]



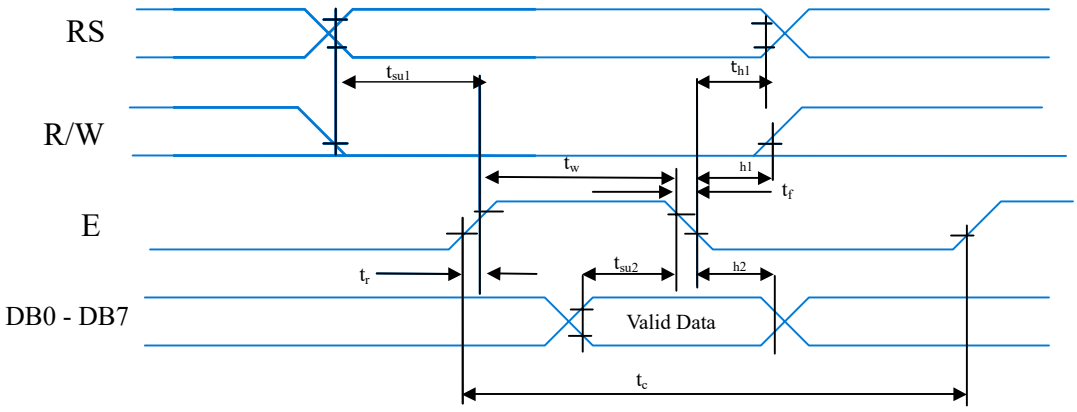
Obr. 3.2 Znázornění zapojení sběrnice IIC [6]

Komunikaci na sběrnici zahajuje master, vysláním hodinového signálu po sběrnicovém vodiči SCL, za tímto signálem následuje vyslání start bitu po datovém vodiči SDA. Jakmile je vyslán start bit následuje adresa zařízení s kterým se bude komunikovat. Nakonec se odešlou data, následované stop bitem, při kterém dojde k ukončení komunikace. [9]

3.3 Paralelní sběrnice

Paralelní sběrnice se hojně využívá v širokém spektru aplikací. Převážně se jedná o aplikace ve kterých je potřeba přenášet objemné množství dat ve velmi krátkých časových intervalech. Nejčastěji se využívá v integrovaných obvodech nebo pro řadiče LCD zobrazovačů. Tento způsob komunikace se spíše využívá pro velké zobrazovací plochy, jako jsou televizní obrazovky a monitory. Sběrnice se objevuje v různé bitové velikosti od nejmenších 4 bitových, až po velké paralelní sběrnice dosahující i 24 nebo více bitů. S tím je ovšem spojená nevýhoda při zvyšování počtu bitů u paralelní sběrnice se tím zvyšuje i počet potřebných vodičů mezi zařízeními. Jeden vodič odpovídá jednomu přenesenému bitu.

Přenášení dat po paralelní sběrnici je znázorněno na obrázku 3.3. Jedná se o průběh komunikace u řadiče ST7066. Nejdříve se provede nastavení signálů R/W a RS. První signál nastavuje, zda se jedná o akci čtení či zápis, v našem případě se jedná o zápis dat do řadiče. Signál RS označuje, zda se přenáší data či příkazy. Následně dojde k nastavení vstupu E do logické jedničky a na paralelní bránu se přivedou logické signály, které chceme přenést do řadiče. Jakmile se objeví sestupná hrana na vstupu E, řadič začne číst bitové hodnoty, které jsou na paralelní sběrnici. Pro zajištění správného přečtení signálů je nutné, aby přivedené signály na paralelní sběrnici zůstali ustálené a neměnné po celou dobu čtení.

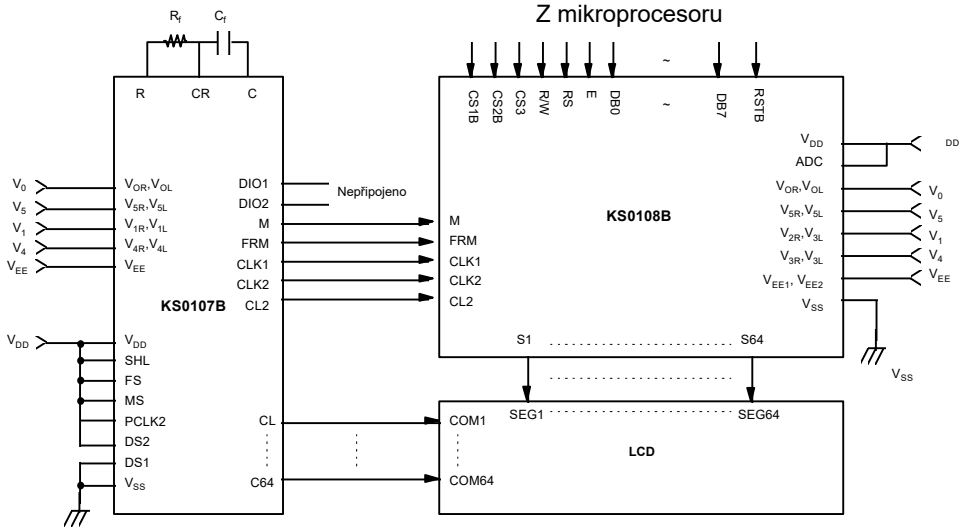


Obr. 3.3 Znázornění cyklu pro zápis dat přes paralelní bránu.
[2]

4 Řadiče pro černobílé zobrazovače

4.1 Řadič KS0108

Řadič KS0108B umožňuje spojování řadičů do kaskád. Řadič má vyvedenou vlastní sběrnici, která umožňuje spojení více řadičů mezi sebou. Velkou výhodu je připojení jen jednoho řadiče z celé kaskády k mikroprocesoru. Řadič KS108B komunikuje s mikrokontrolérem a má k dispozici 64 výstupních pinů pro řízení sloupcových signálů displeje. Pro řízení řádkových signálů je k řadiči připojen další řadič KS0107B, propojení je realizováno přes vlastní sběrnici. Zapojení je znázorněno na obrázku 4.1. Řadič má k dispozici interní RAM paměť pro ukládání zobrazovaných dat, které byly přeneseny přes 8 bitovou sběrnici s mikrokontroléru. Na výstupní sběrnici posílá výstupní signál, který koresponduje s daty uloženými v paměti RAM. Řadič je určen pro řízení maticových displejů využívajících technologii tekutých krystalů. [15]

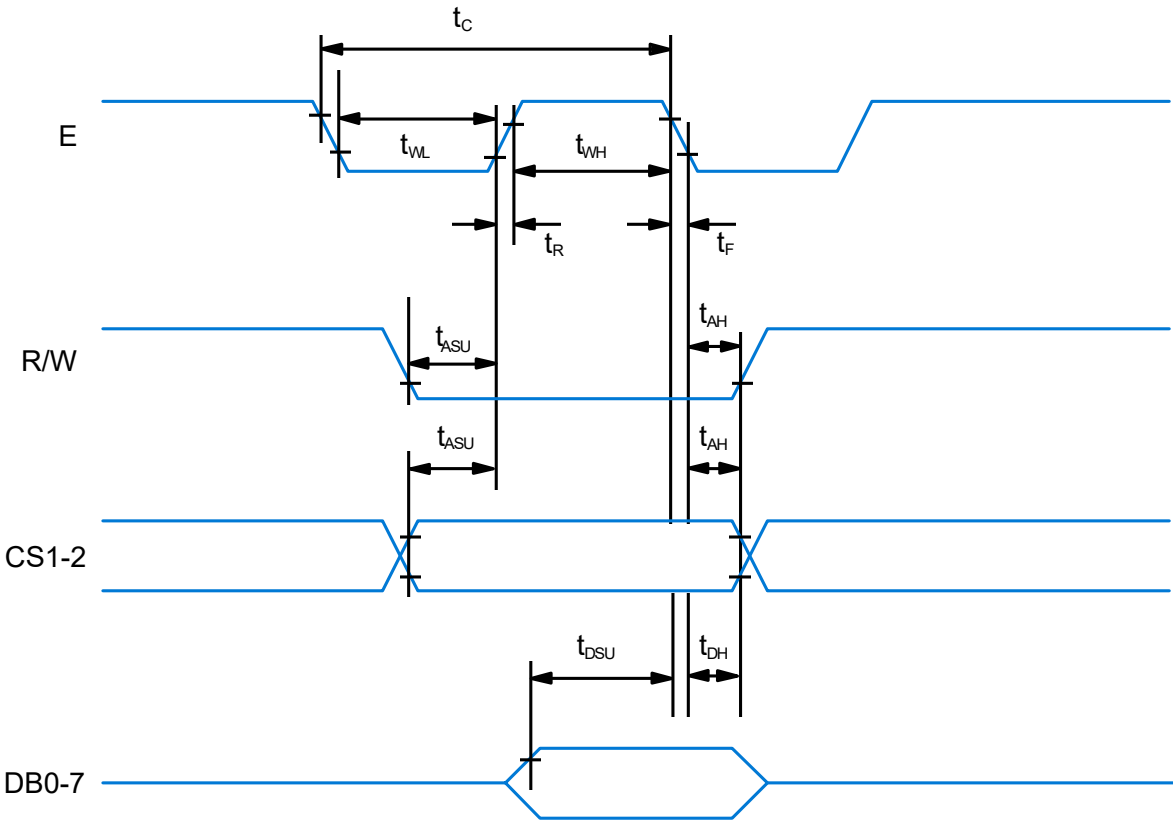


Obr. 4.1 Schéma spojování řadičů KS0108 a KS0107 [15]

Hlavní výhodou těchto dvou řadičů je možnost spojování do kaskád, nemusí se jednat o spojení jen dvou řadičů. Můžeme spojovat i více těchto řadičů a dosáhnout větší zobrazované oblasti a zároveň komunikovat jen po jedné sběrnici. Mezi další přednosti tohoto řadiče patří 8 bitová paralelní sběrnice, paměťový prostor RAM o celkové velikosti 512 bytů (4096 bitů). Kdy uložená logická 1 v paměti představuje, že daný pixel je aktivní. Vstupní napájecí napětí musí být 5 V. Spotřeba řadiče dle datasheetu je během zobrazování pixelů kolem 100 uA, při měření bez zátěže. Při komunikaci se proudová špička dostane až na 500 uA, opět bez zátěže. Bez zátěže představuje situaci, kdy k řadiči není připojen žádný zobrazovač. [15]

4.1.1 Délka zápisového impulzu

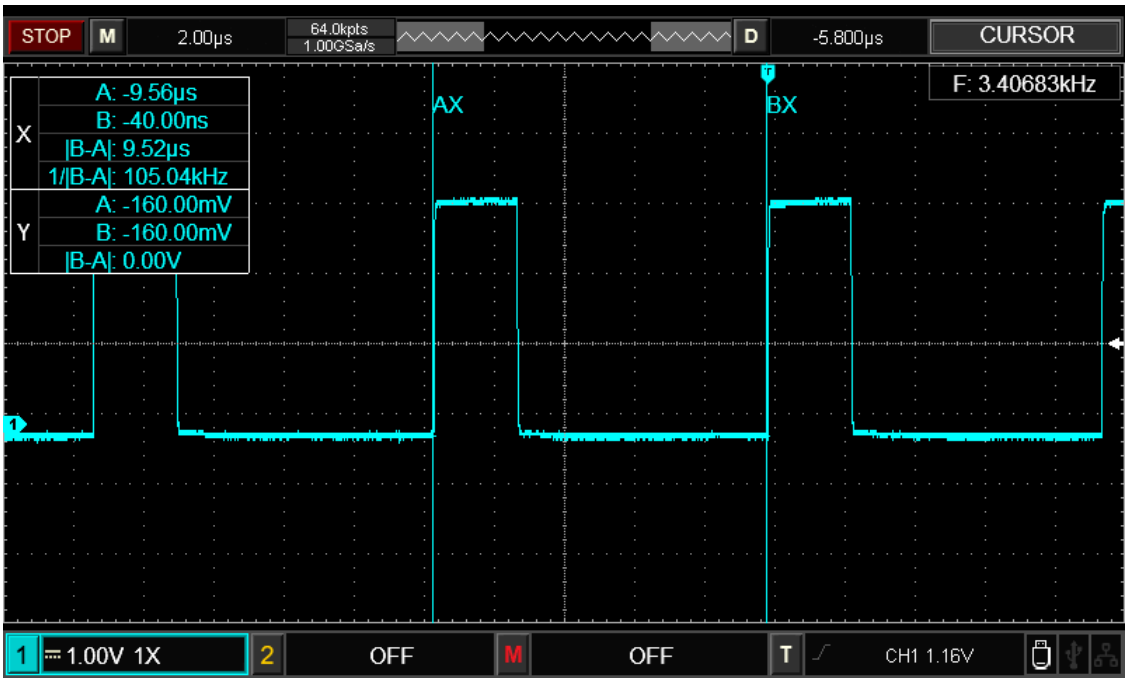
Při použití řadiče musíme dbát na minimální délku zápisového impulzu. Řadič reaguje na vzestupnou hranu impulzu E. Přivedení impulzu způsobí, že řadič si přečte signály přivedené na datové piny D0 - D7. Průběh tohoto impulzu včetně datových signálů je znázorněn na obrázku 4.2. Modře jsou znázorněny průběhy signálů přivedené do řadiče. Minimální délky časů t_{WL} a t_{WH} musí být větší něž 0,450 us a znázorněná hodnota t_C by měla dosahovat minimálně 1 us. Tato podmínka je nutná při pokusech o zkrácení tohoto impulzu, protože jinak dochází ke ztrátě přenášených dat a řadič se chová nepředvídatelně.



Obr. 4.2 Průběh signálů pro zapisování dat do řadiče KS0108 [15]

Měření délky zápisového impulzu

Měření délky zápisového impulzu jsem provedl pomocí osciloskopu. Připojil jsem osciloskop na výstupní pin E na desce displeje. Prvním měřením jsem zjistil, že celková délka hodnoty t_C byla 14,5 us. Jelikož minimální délka je 1 us, provedl jsem úpravu kódu v podobě zkrácení čekacího intervalu. Aktuálně se jedná o čekací interval o délce jednoho příkazu "nop". Díky tomu se zkrátil interval na 9,52 us. Celé měření jsem uskutečnil na frekvencí procesoru 16 MHz. Změřený průběh osciloskopem je znázorněn na obrázku 4.3.



Obr. 4.3 Změřený průběh zapisovacího impulzu E na řadiči KS0108 [15]

Testoval jsem i variantu pomocí využíti interního čítače a přerušení. Tato metoda se ovšem neosvědčila jako vhodná. Jelikož byla velice pomalá oproti použití příkazu "nop". Ovšem jedinou nepatrnou výhodou byla dynamická změna čekacího času v závislosti na frekvenci procesoru.

4.1.2 Zapojení desky

Tabulka 4.1 znázorňuje dvě různé varianty v provedení vývodů na desce s osazeným řadičem KS0108 a zobrazovačem. Lze vidět odlišnosti v zapojení desky, což způsobuje problémy v kompatibilitě při přechodu mezi různými verzemi. Testoval jsem desku ve verzi 2.

4.1.3 Reset řadiče

Ke správné funkci řadiče a zobrazování dat na LCD displeji je nejdříve potřeba provést správnou inicializaci. Postup inicializace je znázorněn ve výpisu kódu 4.1, kde se nejprve provede nastavení výstupní brány u mikrokontroléru. Předtím než je zahájen přenos dat do řadiče, je nejprve nutné provést reset řadiče.

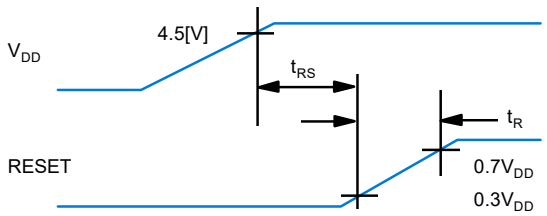
V případě neprovedení resetu, ale připojení výstupního pinu reset přímo na napájecí napětí, v tomto případě na 5 V, dojde k občasné chybě, kdy se na displeji začnou zobrazovat náhodné pixely a znaky. Chyba je způsobena signálem reset, který musí přecházet z logické 0 do 1, až poté co napájející napětí dosáhne 5 V a po uplynutí 1 us. Průběh je modře znázorněn na obrázku 4.4 a hodnoty v tabulce 4.2. [15]

Tab. 4.1 Porovnání vývodů u jednotlivých verzí desky

Číslo pinu	Komunikace	
	Verze 2	Verze 1
1	CS1	GND
2	CS2	VCC
3	GND	VEE
4	VCC	RS
5	VEE	R/W
6	RS	E
7	R/W	DB0
8	E	DB1
9	DB0	DB2
10	DB1	DB3
11	DB2	DB4
12	DB3	DB5
13	DB4	DB6
14	DB5	DB7
15	DB6	CS1
16	DB7	CS2
17	RESET	RESET
18	VOUT	VOUT
19	LED+	LED+
20	LED-	LED-

Tab. 4.2 Minimální délka náběžné hrany resetu po zapnutí napájení

Název	Symbol	Minimální hodnota	Maximální hodnota	Jednotky
Čas Resetu	t_{RS}	1,0	-	us
Čas vzestupné hrany	t_r	-	200	ns



Obr. 4.4 Průběh reset signálu po zapnutí napájecího napětí [15]

Předcházení tohoto problému je provedení řádného resetu, jak je znázorněno v kódu 4.1 na řádku 17 až 21. Po provedení resetu se do řadiče odešlou inicializační data a následně zobrazovací data, která obsahují samé nuly. Tudíž se po inicializaci nic na displeji nezobrazí.

```
1 void Init_KS0108(void) {
2   GPIO_Set(DATA_0, Port_OutputPP);
3   GPIO_Set(DATA_1, Port_OutputPP);
4   GPIO_Set(DATA_2, Port_OutputPP);
```

```
5    GPIO_Set(DATA_3, Port_OutputPP);
6    GPIO_Set(DATA_4, Port_OutputPP);
7    GPIO_Set(DATA_5, Port_OutputPP);
8    GPIO_Set(DATA_6, Port_OutputPP);
9    GPIO_Set(DATA_7, Port_OutputPP);
10
11   GPIO_Set(RS_PAR, Port_OutputPP);
12   GPIO_Set(CS1_PAR, Port_OutputPP);
13   GPIO_Set(CS2_PAR, Port_OutputPP);
14   GPIO_Set(E_CLK, Port_OutputPP);
15
16   GPIO_Set(RESET, Port_OutputPP);
17   GPIO_Write(RS_PAR, 0);
18   WaitMs(500);
19   GPIO_Write(RS_PAR, 1);
20   WaitMs(1000);                    //chvilku počkej než se načte LCD po spuštění
21
22   GPIO_Write(RS_PAR, 1);
23   GPIO_Write(CS1_PAR, 0);
24   GPIO_Write(CS2_PAR, 0);
25   GPIO_Write(E_CLK, 0);
26
27   //LCD ON
28   KS0108_Send_Command(0x3F, 0);
29   KS0108_Send_Command(0x3F, 1);
30
31   KS0108_Fill(0x00);
32   VRAM_Send_KS0108();
33 }
```

Výpis kódu 4.1 Inicializace řadiče KS0108

4.1.4 Ukázka kódu

Pro posílání dat do řadiče, přes 8-bitovou paralelní sběrnici, jsem využil výstupní bránu GPIOB u mikrokontroléru. Posílání dat je znázorněno v kódu 4.2. Vynuluje se výstupní brána, následně se na ní zapíšou data, která se budou posílat. Protože dostupný displej disponuje dvěma řadiči KS0108 na řádku 7 - 10 provedu výběr řadiče. Každý řadič dovoluje zobrazovat na ploše 64 x 64 pixelů. Pro posílání příkazů do řadiče jsem využil jinou funkci, která je v podstatě stejná, jen je rozdíl na řádku číslo 5 v kódu 4.2. Rozdíl spočívá v nastavení signálu RS_PAR_Pin na logickou 0 a pro data je tento pin nastaven na logickou 1.

```
1  static void KS0108_Send_Data(uint8_t com, bool cs) {
2    //Reset GPIOB DATA
3    GPIOB->BSRR = 0x00FF0000;
4    GPIOB->BSRR = com;
5    GPIOB->BSRR = (1 << (RS_PAR_Pin));    //RS = 1 for DATA
6
7    if(cs == 0) {
8      GPIOB->BSRR = ((1 << (CS1_PAR_Pin)) | ((1 << (CS2_PAR_Pin)) << 16));
9    }else{
10     GPIOB->BSRR = ((1 << (CS2_PAR_Pin)) | ((1 << (CS1_PAR_Pin)) << 16));
11   }
12
13   GPIOB->BSRR = (1 << (E_CLK_Pin));            //E Enable
14   Timer_Wait();
15   GPIOB->BSRR = ((1 << (E_CLK_Pin)) << 16);      //E Distable
16   Timer_Wait();
17 }
```

Výpis kódu 4.2 Posílání dat do řadiče KS0108

4.1.5 Ukázka výsledku

Na obrázku 4.5 je vidět ukázka jedné s testovací funkce, která je součástí knihovny.

Obr. 4.5 Ukázka výsledku na displeji s řadičem KS0108

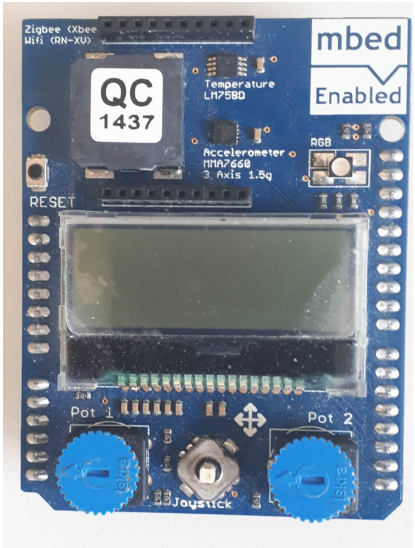
4.2 Řadič ST7565

Jedná se o řídící obvod pro maticové černobílé LCD obrazovky. Umožňuje řízení displeje až do rozlišení 135 x 65 pixelů. Pro komunikaci s mikrokontrolérem má k dispozici 8 bitovou paralelní sběrnici nebo lze využít 4-vodičovou SPI sběrnici. Kapacita integrované RAM paměti je 8580 bitů, kde každý jeden bit představuje 1 pixel na obrazovce. Spotřeba řadiče udávaná v datasheetu při napájení 3,3 V se v normálním módu pohybuje okolo 90 až 130 uA. Po zapnutí vysoko výkonového módu se odebíraný proud vyšplhá na 190 uA. Každý tento čip je vybaven vlastním RC oscilátorem. Lze využít low-power mód pro snížení spotřeby v podobě vypnutí zobrazovaných dat na displeji.

Oproti již zmíněnému řadiči KS0108 popsán v bodě 4.1, má tento řadič výhodu, že se jedná o samostatný řadič. ST7565 zajišťuje řízení řádkových i sloupcových signálů samostatně. To ovšem neznamená, že obvody nelze spojovat do kaskád, ale řadič nemá k dispozici speciální sběrnici pro spojování těchto řadičů. Pro spojení je potřeba připojit další vodič pro výběr daného řadiče. Spojením více stejných řadičů do kaskád přináší výhodu ohledně zobrazovaní na větší plochu.

4.2.1 Mbed Shield

Řadič včetně displeje jsem měl k dispozici na MBed Shieldu, kde byla vyvedená jen sběrnice SPI, tudíž nešlo zvolit jiný způsob komunikace. Více informací o MBed Shieldu lze dočíst zde [14]. Celá deska Mbed Shieldu je vyobrazena na obrázku 4.6.



Obr. 4.6 Vyobrazení Mbed shieldu

4.2.2 Ukázka kódu

Následující úsek kódu ukazuje inicializaci řadiče ST7565. Tento kód jsem dostal v rámci předmětu Programování mikrokontrolérů a nebyl důvod ho přepisovat, navíc umožňuje využívat posílání dat do řadiče za pomocí DMA. Což spočívá v posílání dat bez zásahu mikrokontroléru. Pouze jsem upravil inicializaci SPI sběrnice a nahradil jí univerzálnější funkcí. První parametr funkce určuje používanou sběrnici SPI a druhým parametrem funkce je mód SPI sběrnice.

```
1  void Init_ST7565() {
2    Init_SPI(SPI_1, SPI_Mode3);
3
4  #if defined (ST7565_DMA)
5    Init_DMA_ST7565();
6  #endif
7
8    ST7565_Reset();
9
10   ST7565_Send(0xAE, 0);    //  display off
11   ST7565_Send(0xA2, 0);    //  bias voltage
12   ST7565_Send(0xA0, 0);
13   ST7565_Send(0xC8, 0);    //  colum normal
14
15   ST7565_Send(0x22, 0);    //  voltage resistor ratio
16   ST7565_Send(0x2F, 0);    //  power on
17   //wr_cmd(0xA4);        //  LCD display ram
18   ST7565_Send(0x40, 0);    // start line = 0
19   ST7565_Send(0xAF, 0);      // display ON
20
21   ST7565_Send(0x81, 0);    //  set contrast
22   ST7565_Send(0x17, 0);    //  set contrast
23
24   ST7565_Send(0xA6, 0);      // display normal
25 //  ST7565_Send(0xA7, 0);      // display inverted
26 //  ST7565_Send(0xa5, 0);
27 }
```

Výpis kódu 4.3 Inicializace řadiče ST7565

4.2.3 Ukázka výsledku

Na následujícím obrázku 4.7 je zobrazena ukázka výsledku. Na displeji se zobrazuje Mandelbrotova množina.

Obr. 4.7 Ukázka zobrazovaného výsledku

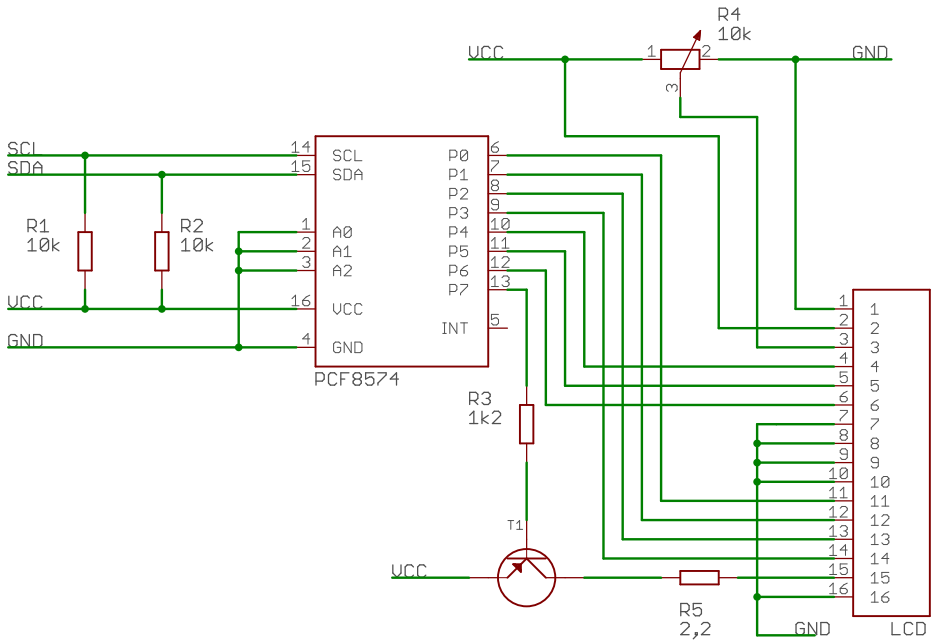
4.3 Řadič ST7066

Alfanumerické zobrazovače nezvládají zobrazovat žádné grafické výstupy. Zvládají jen znaky v podobě číslic, písmen a speciální znaky (závorka, lomítko, atd..). K ukládaní těchto znaků slouží vnitřní ROM paměť o velikosti 13200 bitů, kde je z výroby nahráno 240 znaků v rozlišení 5 x 8 bodů. Někteří výrobci volí i EEPROM paměť, v případě potřeby lze poslat řadič zpět výrobci, který změní obsah paměti na přání zákazníka. Ovšem tento proces není zdarma. Je možné využít i druhou variantu pro vytváření vlastních znaků. Znakovou základnu můžeme rozšířit o vlastní znaky, které se uloží do vnitřní RAM paměti. Ovšem kapacita paměti dovoluje uchovat maximálně 8 volitelných znaků o velikosti 5 x 8 pixelů nebo 4 znaky o 5 x 10 bodů. Nevýhoda použití tohoto způsobu je ztráta uložených znaků po vypnutí napájení. Komunikace s řadičem je zajištěna pomocí 4 bitové nebo 8 bitové paralelní sběrnice. Řadič zvládne zobrazovat do velikosti 8 x 80 bodů. Přepočtem na znaky to jsou dva řádky, každý řádek má 8 znaků. Napájecí napětí musí být 5 V, pokud je nižší než 4,5 V řadič se ani nespustí. Spotřeba řadiče uvedená v datasheetu je mezi 0,2 až 0,5 mA, kdy je spotřeba měřena naprázdno.[2]

Na rozdíl od grafických řadičů se u alfanumerických řadičů nevyužívá funkce bufferu, kdy se zobrazované data ukládaly do RAM paměti v mikrokontroléru a následně nahrávali jako celý snímek. Zde je využito přímého zápisu do paměti řadiče. S tímto řešením se pojí problém v podobě nutnosti nejdříve odeslat pozici pro zobrazovaná data. Tedy řádek a sloupec kde se začne zapisovat do paměti. Vytvořený kód jsem testoval na řadičích HD44780 a ST7066. Oba řadiče mají stejný příkazy i inicializaci jen řadič HD44780 má o něco menší vnitřní ROM paměť, která dosahuje jen 9920 bitů. Dovoluje zobrazit 208 znaků v rozlišení 5 x 8 pixelů. Což je na úkor velkých znaků, kterých zvládá jen 32 v rozlišení 5 x 10 pixelů. Výhodou řadiče HD44780 je že má k dispozici 4 řádky po 20 znacích. Zatím se mi nepodařilo vyřešit adresování na zbylé dva řádky. Aktuálně je knihovna funkční jen pro první dva řádky. [11]

4.3.1 Využití IIC

Pro vyzkoušení jiného způsobu komunikace než 8 bitové paralelní brány jsem použil IIC převodník s integrovaným obvodem PCF8574. Jedná se o sérioparalelní převodník, který dovoluje pomocí 2 sériových vodičů ovládat 8 paralelních. Zapojení převodníku k LCD displeji je znázorněno ve schématu 4.8. Převodník je připojen pomocí 4 datových vodičů k displeji na vývody D4 - D7. Další 3 vývody jsou připojeny k řídicím signálům RS, RW, EN. Poslední vývod přes tranzistor na řízení podsvícení LCD displeje. Více informací o tomto čipu lze najít v datasheetu [1].



Obr. 4.8 Zapojení displeje s IIC převodníkem PCF8474

4.3.2 Ukázka kódu

Znázorněný kód 4.4 ukazuje posílání dat do řadiče HD44780 s využitím IIC sběrnice. Nejdříve je provedeno rozložení 16 bitového slova po jednotlivých 4 bitech a nastavení signálů EN a RS. Kde EN přestavuje zápisový impulz a hodnota RS představuje práci s instrukčním registrem.

```
1  static void ST7066_Send_Data(char data) {
2    char data_u, data_l;
3    uint8_t data_t[4];
4    data_u = (data & 0xf0);
5    data_l = ((data << 4) & 0xf0);
6    data_t[0] = data_u | 0x0D;                    //en=1, rs=0
7    data_t[1] = data_u | 0x09;                    //en=0, rs=0
8    data_t[2] = data_l | 0x0D;                    //en=1, rs=0
9    data_t[3] = data_l | 0x09;                    //en=0, rs=0
10   I2CX_Send_Data(SLAVE_ADDRESS_LCD, (uint8_t*) data_t, 4);
11 }
```

Výpis kódu 4.4 Posílání dat do řadiče ST7066

4.3.3 Ukázka výsledku

Ukázkový text zobrazovaný na zobrazovačích s řadiči ST7066 a HD4780U je znázorněn níže.

Obr. 4.9 Ukázka výsledku na displeji s řadičem ST7066

Obr. 4.10 Ukázka výsledku na displeji s řadičem HD4780U

5 Řadiče pro barevné zobrazovače

5.1 Řadič ST7735

Grafický řadič ST7535 slouží pro zobrazování na TFT-LCD obrazovkách. Na rozdíl od všech předcházejících řadičů tento řadič ukládá jeden pixel do 18 bitů, kde každých 6 bitů představuje jednu barevnou složku z RGB. Není potřeba využívat celý rozsah barev, lze si nastavit barevnou hloubku na 12, 16 nebo 18 bitů. Díky tomu zvládá zobrazit až 262 tisíc barev. Barvy jsou reprezentovány pomocí barevných složek RGB. Řadič disponuje více rozhraní, k dispozici má několik paralelních sběrnic 8/9/16/18 bitové, ze sériových sběrnic lze využít třívodičovou nebo čtyřvodičovou sběrnici SPI. Maximální rozlišení řadiče je 132 x 162 pixelů. Velikost interní RAM paměti je 132 x 162 x 18 bitů. Napájecí napětí musí být 3,3 V. Spotřeba řadiče je nezávislá na zobrazovaných datech. V datashetu jsou uvedeny spotřeby při zobrazení všech černých nebo všech bílých pixelů v hodnotě odebíraného proudu 0,01 mA. Během komunikace odebíraný proud dosahuje až 0,5 mA, nezávisle na zobrazované informaci na displeji. Data jsou uváděná bez podsvícení. S přidáním podsvícení vzroste odběr displeje až na 50 mA. Řadič má řadu výhod, částečně programovatelné provozní cykly, nastavitelné barevné hloubky, řádkové inverze a inverze celého obrazce. [3]

5.1.1 Ukázka kódu

Odesílání kódu do řadiče ST7735 je znázorněno ve výpisu kódu 5.1. Nejdříve je provedeno nastavení jednotlivých výstupních pinů a inicializace sériové sběrnice SPI. Následně je řadič probuzen ze spánku, není potřeba probouzet řadič ihned na začátku inicializace, protože většinu příkazů lze úspěšně provést i v uspaném režimu ovšem ne všechny. Nastavením 16 bitového barevného formátu a použiji jednu z předvolených nastavení gamma korekce. Gamma korekce si lze nastavit i vlastní hodnoty, nastavení vlastních hodnot gamma korekcí není vyobrazeno ve výpisu kódu. Následně je nastavena funkce displeje, počet snímku, formát barev, napájecí obvody, výběr zobrazované oblasti. Formát barev představuje zda využiji volbu RGB nebo GBR. Zde je využito RGB. Předposledním příkazem je povolení zápisu do RAM paměti a poslední je provedeno nastavení rotace displeje, která je blíže vyobrazena v části 5.1.2. Na konci inicializace je provedeno první nahrání dat do displeje, kde se na displeji zobrazí černé pozadí, výchozí nastavení po resetu je bílé pozadí, takže je i vidět kdy je inicializace dokončena.

```
1 void Init_ST7535() {
2   GPIO_Set(PIN_RSTN, Port_OutputPP);
3   GPIO_Write(PIN_RSTN, 1);
4   GPIO_Set(PIN_CS, Port_OutputPP);
5   GPIO_Write(PIN_CS, 1);
6   GPIO_Set(PIN_DC_A0, Port_OutputPP);
7   GPIO_Write(PIN_DC_A0, 1);
```

```
8
9   Init_SPI(SPI_1, SPI_Mode0);
10  ST7535_Send_Command(CMD_SLEEPOUT);          //exit sleep
11  ST7535_Send_Command(CMD_PIXFMT);            //Set Color Format 16bit
12  ST7535_Send_Data_8(0x05);
13  ST7535_Send_Command(CMD_GAMMASET);          //default gamma curve 3
14  ST7535_Send_Data_8(0x04);                   //0x04
15
16  ST7535_Send_Command(CMD_NORML);
17  ST7535_Send_Command(CMD_DFUNCTR);
18  ST7535_Send_Data_8(0b11111111);
19  ST7535_Send_Data_8(0b00000110);
20
21  ST7535_Send_Command(CMD_FRMCTR1);           //Frame Rate Control (In normal
      mode/Full colors)
22  ST7535_Send_Data_8(0x08);                   //0x0C//0x08
23  ST7535_Send_Data_8(0x02);                   //0x14//0x08
24
25  ST7535_Send_Command(CMD_DINVCTR);           //display inversion
26  ST7535_Send_Data_8(0x07);                   //0x07
27
28  ST7535_Send_Command(CMD_PWCTR1);            //Set VRH1[4:0] & VC[2:0] for VCI1
      & GVDD
29  ST7535_Send_Data_8(0x0A);                   //4.30 - 0x0A
30  ST7535_Send_Data_8(0x02);                   //0x05
31
32  ST7535_Send_Command(CMD_PWCTR2);            //Set BT[2:0] for AVDD & VCL & VGH
      & VGL
33  ST7535_Send_Data_8(0x02);
34
35  ST7535_Send_Command(CMD_VCOMCTR1);          //Set VMH[6:0] & VML[6:0] for VOMH
      & VCOML
36  ST7535_Send_Data_8(0x50);                   //0x50
37  ST7535_Send_Data_8(99);                     //0x5b
38
39  ST7535_Send_Command(CMD_VCOMOFFS);
40  ST7535_Send_Data_8(0);                      //0x40
41
42  ST7535_Send_Command(CMD_CLMADRS);           //Set Column Address
```

```
43    ST7535_Send_Data_16(0x00 + OFFSET_X);
44    ST7535_Send_Data_16(Pix_Collums + OFFSET_X);
45
46    ST7535_Send_Command(CMD_PGEADRS);              //Set Page Address
47    ST7535_Send_Data_16(0x00 + OFFSET_Y);
48    ST7535_Send_Data_16(Pix_Rows + OFFSET_Y);
49
50    ST7535_Send_Command(CMD_DISPON);               //display ON
51    ST7535_Send_Command(CMD_RAMWR);                //Memory Write
52
53  ...
54    ST7535_Fill(0x0000);
55  }
```

Výpis kódu 5.1 Posílání dat do řadiče ST7735

5.1.2 Nastavení orientace displeje

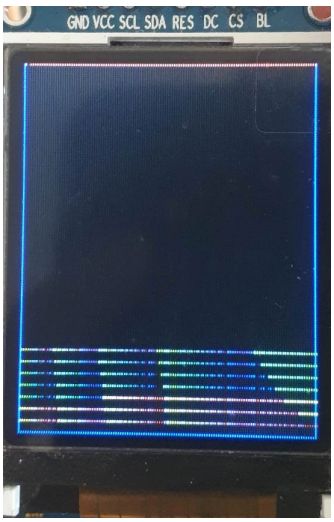
Na některých řadičích lze nastavit s jakým otočením se budou data zobrazovat na displeji. Což je celkem praktické v případě, kdy máme displej umístěný v krabičce a tu nelze otočit, tak provedeme jednoduché softwarové otočení, které je implementováno v řadiči. Nemusíme pro tento případ vymýšlet vlastní způsob otáčení dat v RAM paměti. Způsob otáčení je znázorněn v bloku kódu 5.2. Otáčení dat podporují řadiče ST7735, IL9163, GC9A01 a ST7789. Obecně lze říci, že většina řadičů, které jsou určeny pro barevné zobrazovací plochy.

```
1   ...
2     ST7535_Send_Command(CMD_MADCTL);
3
4   #if ORIENTATION == 0
5     ST7535_Send_Data_8(0x18);
6   #elif ORIENTATION == 1
7         ST7535_Send_Data_8(0x28);
8     #elif ORIENTATION == 2
9         ST7535_Send_Data_8(0x48);
10    #else
11        ST7535_Send_Data_8(0x88);
12    #endif
13  ...
```

Výpis kódu 5.2 Orientace zobrazovaných dat

5.1.3 Ukázka výsledku

Na obrázku 5.1 je znázorněna ukázka zobrazení dat na displeji s řadičem ST7735. Lze vidět zobrazovaní horizontálních čar s měnící se barvou.



Obr. 5.1 Ukázka výsledku na displeji s řadičem ST7735

5.2 Řadič IL9163

Řadič IL9163 slouží pro řízení barevných TFT displejů s tekutými krystaly. Rozlišení displeje je 132 x 162 pixelů, zobrazovaná data jsou uloženy v interní RAM paměti, která má kapacitu 48 114 bytů. Komunikaci s řadičem lze zajistit pomocí 8/9/16/18 bitové paralelní sběrnice nebo sériové sběrnice SPI. Řadič je vybaven užitečnými vlastnostmi přímo na čipu mezi které patří časový generátor, oscilátor, DC/DC převodník, 8 přednastavených gamma křivek, řádková/obrazová inverze. Řadič je napájen 3,3 V a celkový odebíraný proud řadiče uváděný v datasheetu je kolem 1 mA. [7]

Celkové vlastnosti jsou velice podobné řadiči ST7735, viz. 5.1. Největším rozdílem je velikost zobrazované plochy, kterou řadiče podporují. Následně řadič IL9163 je vybaven o něco více příkazy než již zmíněný řadič ST7735. Nejedná se o zásadní rozdíl, jde o příkazy pro nastavení posunu zobrazovaného prostoru.

5.2.1 Ukázka kódu

Celkový obsah kódu je velice podobný řadiči ST7735, odlišnosti jsou pouze v jedné části, kterou jsem ve výsledné práci nevyužil. Odlišnost spočívá v nastavení posunu prvního řádku a sloupce. Posun je potřeba provádět v případech, kdy výrobce špatně napáruje příslušné řádky nebo sloupce vzhledem k RAM paměti řadiče. Tento jev se převážně vyskytuje u řadičů určených pro větší zobrazované plochy než je připojená obrazovka. Kdy všechny řádkové vodiče nemusí být připojené k obrazovce a dochází k posunu obrazu mezi RAM pamětí a zobrazovanou plochou.

K řešení posunu je možno využít zmíněnou funkci, kterou lze vidět ve výpisu kódu 5.3. Funkce nastavuje nové místo počátku v RAM paměti řadiče. Tento proces nastavení je nutné provést po každém zapnutí napájecího napětí. Ovšem tuto funkci jsem nepoužil.

```
1 #if defined (IL9163)
2   // Jen pro IL9163
3   ST7535_Send_Command(CMD_VSCLLDEF);
4   ST7535_Send_Data_16(0x00 + OFFSET_X);
5   ST7535_Send_Data_16(Pix_Collums);
6   ST7535_Send_Data_16(0x00 + OFFSET_Y);
7 #endif
```

Výpis kódu 5.3 Nevyužitá funkce na offset

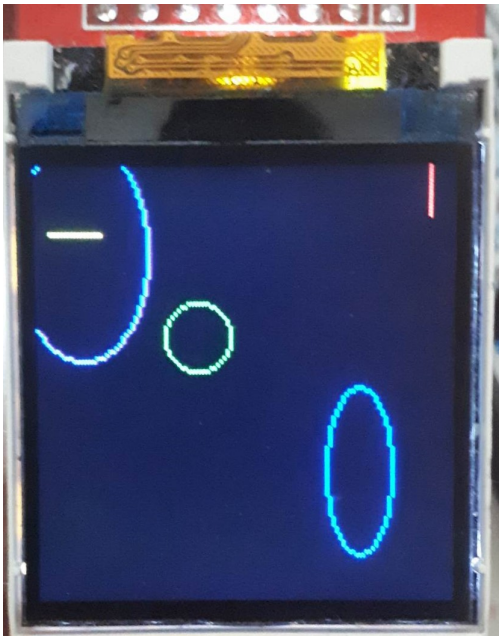
Využil jsem ručního výběru oblasti do které se mají nahrávat zobrazovaná data, část výběru je zobrazena ve výpisu kódu 5.4. Jedinou výhodou této metody je její rozšířenost mezi řadiči určenými pro barevné zobrazovače. Princip je podobný jako výše zmíněná funkce.

```
1  ST7535_Send_Command(CMD_CLMADRS);       // Set Column Address
2  ST7535_Send_Data_16(0x00 + OFFSET_X);
3  ST7535_Send_Data_16(Pix_Collums + OFFSET_X);
4
5  ST7535_Send_Command(CMD_PGEADRS);       // Set Page Address
6  ST7535_Send_Data_16(0x00 + OFFSET_Y);
7  ST7535_Send_Data_16(Pix_Rows + OFFSET_Y);
```

Výpis kódu 5.4 Řešení offsetu

5.2.2 Ukázka výsledku

Ukázka zobrazení je znázorněna na obrázku .



Obr. 5.2 Ukázka výsledku na displeji s řadičem IL9163

5.3 Řadič GC9A01

Řadič GC9A01 slouží pro řízení barevných TFT displejů, který dovoluje zobrazit až do rozlišení 240 x 240 pixelů. Pro funkci řadiče nejsou potřeba žádné externí komponenty. Komunikace je zajištěna pomocí klasických sběrnic, podobně jako u řadičů ST7735 nebo IL9163. K dispozici má sériovou SPI sběrnici a dále i paralelní sběrnici 6/9/12/16/18 bitovou. Řadič je napájen 3,3 V a maximální proudový odběr je nižší než 48 mA, hodnota je udávaná s podsvícením displeje. [12]

Specialitou tohoto displeje je, že se nejedná o klasický čtvercový nebo obdélníkový displej, ale o kulatý zobrazovač. Aby bylo možné zobrazovat data na kulatém displeji, tak se některé oblasti v paměti nezobrazují. Největší obrazec lze zobrazit uprostřed displeje. Úhlopříčka displeje má velikost 1,28 inch (3,2512 cm) [12].

5.3.1 Ukázka kódu

Nejzajímavější části kódu pro tento řadič je inicializace. Pro posílání dat do řadiče fungují stejné funkce, jako již pro zmíněné řadiče ST7735 a IL9163 o těchto řadičích je zmíněno v kapitolách 5.1 a 5.2.

Inicializace řadiče začíná stejně jako u řadiče ST7735, kde je nastavena výstupní brána a konfigurace sběrnice SPI. Obdobně je zde nakonfigurováno i nastavení hloubky barev, gamma funkce a napájecích obvodů. Zajímavější část inicializace pokračuje na řádku 71 ve výpisu kódu 5.5. Ve vyobrazené části inicializačního kódu je velice náročné zjistit, co vlastně následující příkazy vykonávají. Jelikož tyto příkazy nejsou uvedeny v datasheetu řadiče. Při pokusech odstranit tuto část kódu nebo ji nahradit čekacím intervalem, se zobrazovač začne chovat zvláštně a zobrazuje pouze náhodné vertikální čáry vedoucí přes celou obrazovku a k tomu ještě problikává. Po této části následuje zapnutí displeje a nastavení rotace zobrazovaných dat. Posledním příkazem je poslání černého pozadí na celý displej, což je totožné jako u ostatních řadičů.

Řešením problikávání displeje a vertikálních čar bylo přidání části následujících příkazů zobrazených ve výpisu kódu 5.5, jedná se o úryvek z inicializační funkce. Ovšem není známo co vlastně tyto příkazy vykonávají, jelikož nejsou uvedené v datasheetu. Tímto řešením jsem se inspiroval již vzniklým kódem. [4]

```
67 ...
68   ST7535_Send_Command(CMD_FRAMERATE);
69   ST7535_Send_Data_8(0x34);
70   WaitMs(500);
71
72   ST7535_Send_Command(0x62);
73   ST7535_Send_Data_8(0x18);
74   ST7535_Send_Data_8(0x0D);
75   ST7535_Send_Data_8(0x71);
```

```
76    ST7535_Send_Data_8(0xED);
77    ST7535_Send_Data_8(0x70);
78    ST7535_Send_Data_8(0x70);
79    ST7535_Send_Data_8(0x18);
80    ST7535_Send_Data_8(0x0F);
81    ST7535_Send_Data_8(0x71);
82    ST7535_Send_Data_8(0xEF);
83    ST7535_Send_Data_8(0x70);
84    ST7535_Send_Data_8(0x70);
85  ...
86    ST7535_Send_Command(0x98);
87    ST7535_Send_Data_8(0x3e);
88    ST7535_Send_Data_8(0x07);
89
90    ST7535_Send_Command(CMD_TEFXLON);
91    ST7535_Send_Command(CMD_DINVON);
92  ...
```

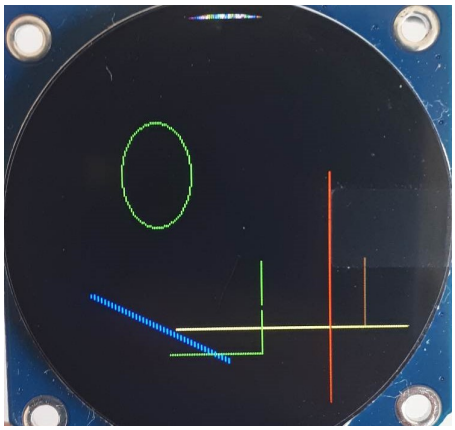
Výpis kódu 5.5 Inicializace řadiče GC9A01

5.3.2 Zobrazovaná oblast displeje

Jak je již zmíněno v části 5.3 jedná o čtvercovou video paměť, která je zobrazována na kulatém displeji. Tudíž ne všechny body jsou zobrazeny na displeji. Mnohem efektivnější řešení je neukládat do RAM paměti mikrokontroléru všechny pixely, ale jen ty, které se zobrazují na displeji. Ovšem poté je i potřeba vyřešit odesílání dat do řadiče, jelikož každý řádek má jiný počet pixelů a bylo by nutné nezobrazované pixely posílat v podobě nulových hodnot. Nebo jinou možností je využít funkci výběru oblasti do které chci zapisovat data a následně je odesílat. Každý řádek se musí odesílat samostatně. Před odesláním dalšího řádku se nejprve vybere oblast následujícího řádku, aby nedocházelo k přepisování předcházejícího řádku.

5.3.3 Ukázka výsledku

Na obrázku 5.3 je vidět zobrazované obrazce na kulatém displeji s řadičem GC9A01.



Obr. 5.3 Ukázka výsledku na displeji s řadičem GC9A01

5.4 Řadič ST7789

Jedná se o samostatný kontrolér pro řízení grafických LCD-TFT zobrazovačů. Tento čip pro komunikaci s mikrokontroléry využívá paralelní sběrnici 8/9/16/18 bitové nebo sériové rozhraní SPI. Maximální velikost zobrazované oblasti je 320 x 240 pixelů. Což odpovídá velikosti RAM paměti, kde každý zobrazovaný pixel představuje 18 bitů v RAM paměti. Celková kapacita RAM paměti je 1 382 400 bitů. Ostatními vlastnostmi se podobá řadiči ST7735. Dovede nastavit různou barevnou hloubku výběrem mezi 12/16/18 bity, Low-Power, programovatelné cykly displeje a čtyřmi přednastavenými gamma funkcemi. Odebíraný proud řadiče včetně displeje ve výchozím nastavení je 6 mA maximálně může dosáhnout 7,5 mA.

5.4.1 Ukázka kódu

Po bližším prozkoumání řadiče jsem dospěl k závěru, že velká část příkazů je velice podobná řadiči ST7735. Tudíž by teoreticky měla fungovat stejná konfigurace. Jelikož jsem neměl tuto teorii možnost otestovat, vytvořil jsem samostatnou inicializační funkci pro tento řadič. Vytvořená funkce je velice podobná jako využívá řadič ST7735, pouze jsem pozměnil nastavení gamma funkce, která je znázorněna v kódu 5.6. Tato korekce gamma funkce byla doporučena datashetem. Je ovšem možné, že bude potřeba provést nějakou úpravu, protože nastavení nebylo testováno.

```
70 ...
71   /*--------------- ST7789 Gamma setting ---------------------------------------*/
72   ST7789_Send_Command(CMD_PGAMMAC);
73   ST7789_Send_Data_8(0xD0);
74   ST7789_Send_Data_8(0x08);
75   ST7789_Send_Data_8(0x11);
76   ST7789_Send_Data_8(0x08);
77   ST7789_Send_Data_8(0x0C);
78   ST7789_Send_Data_8(0x15);
79   ST7789_Send_Data_8(0x39);
80   ST7789_Send_Data_8(0x33);
81   ST7789_Send_Data_8(0x50);
82   ST7789_Send_Data_8(0x36);
83   ST7789_Send_Data_8(0x13);
84   ST7789_Send_Data_8(0x14);
85   ST7789_Send_Data_8(0x2D);
86
87   ST7789_Send_Command(CMD_NGAMMAC);
88   ST7789_Send_Data_8(0xD0);
89   ST7789_Send_Data_8(0x08);
90   ST7789_Send_Data_8(0x10);
```

```
91   ST7789_Send_Data_8(0x08);
92   ST7789_Send_Data_8(0x06);
93   ST7789_Send_Data_8(0x39);
94   ST7789_Send_Data_8(0x44);
95   ST7789_Send_Data_8(0x51);
96   ST7789_Send_Data_8(0x0B);
97   ST7789_Send_Data_8(0x16);
98   ST7789_Send_Data_8(0x14);
99   ST7789_Send_Data_8(0x2F);
100  ST7789_Send_Data_8(0x31);
```

Výpis kódu 5.6 Nastavení gamma korekcí na řadiči ST7789

6 LCD knihovna

Všechny grafické řadiče v knihovně využívají buffer, kromě alfanumerických řadičů. Tento buffer je ukládán do RAM paměti v mikrokontroléru. Díky tomu lze pro zobrazení obrazců na displeji použít libovolnou grafickou knihovnu, která pracuje s bufferem. Samozřejmě je to závislé na použitém displeji, jestliže používáme barevný nebo černobílý displej. Rozdíl je ve velikosti jednoho pixelu v RAM paměti, u černobílého 1 pixel/1 bit a u barevného 1 pixel/18 bitů. Ovšem má to i nevýhodu v podobě omezení velikosti zobrazované plochy v závislosti na velikosti paměti RAM v mikrokontroléru.

6.1 Uspořádání knihovny

Knihovnu jsem se snažil uspořádat do složek po jednotlivých řadičích. Což není vůbec jednoduché, jelikož některé řadiče mají stejnou inicializační sekvenci, ale jsou od jiného výrobce. To ovšem přináší výhodu využít už existující kód pro více řadičů od různých výrobců. Příkladem jsou řadiče IL9163, ST7535 a GCC9A01, všechny tyto řadiče jsou barevné s různým rozlišením, ale mají stejný způsob komunikace a obdobný počet příkazů. Proto jsem tyto řadiče spojil do jediného souboru, aby nebylo nutné vytvářet dva velice podobné soubory s rozdílem kolem deseti řádek kódu. Uspořádání knihovny je znázorněno na obrázku 6.1.

```
LCD Knihovna
├── KS108
├── B_KS108.h ..... Hlavičkový soubor pro řadič KS0108
│   ├── B_KS108_Test.c ..... Hotové ukázkové funkce
│   └── B_KS108.c ..... Hlavní kód řadiče
├── ST7066
│   ├── B_ST7066.h ..... Hlavičkový soubor pro řadič ST7066
│   └── B_ST7066.c ..... Hlavní kód řadiče
├── HD44780
│   ├── B_HD44780.h ..... Hlavičkový soubor pro řadič HD44780
│   └── B_HD44780.c ..... Hlavní kód řadiče
├── ST7535
│   ├── B_ST7535.h ..... Hlavičkový soubor pro řadičů ST7535, IL9163 a GCCA01
│   └── B_ST7535.c ..... Hlavní kód řadičů
├── ST7789
│   ├── B_ST7789.h ..... Hlavičkový soubor pro řadič ST7789
│   └── B_ST7789.c ..... Hlavní kód řadiče
├── ST7565
│   ├── B_ST7565.h ..... Hlavičkový soubor pro řadič ST7565
│   └── B_ST7565.c ..... Hlavní kód řadiče
├── Vram
│   └── Vram.h ..... Nastavení velikosti buffru
└── ReadMe.txt ..... Informace pro práci s knihovnou
```

Obr. 6.1 Uspořádání souborů v knihovně

Pro zjednodušení práce v knihovně jsem použil již existující funkce pro nastavování výstupních bran mikrokontroléru. Dále jsem využil hotové funkce pro komunikaci přes sběrnici SPI a IIC, které jsem si upravil z důvodu správné funkčnosti komunikace. Grafické výstupy jsem generoval také pomocí existující knihovny. Pro testování barevných displejů jsem využil vlastní funkce, které neměli tolik možností, ale jsou dostačující. Všechny tyto použité funkce jsme vytvářeli v rámci cvičení na předmětech "Mikroprocesory a počítače"a "Programování mikrokontrolérů". Všechny funkce jsou zahrnuty v příloze.

6.2 Instalace knihovny

Knihovnu jsem se snažil tvořit tak, aby nebylo náročné s ní pracovat. Pro všechny případy jsem vytvořil následující návod.

Pokud se rozhodnete použít knihovnu je nejprve nutné si ji stáhnou a vložit do vašeho projektu. Součástí knihovny jsou základní funkce pro nastavování pinů, komunikace přes SPI, IIC. Tyto součásti je potřeba také stáhnout a vložit do projektu. Následně je třeba zahrnout všechny složky do projektu, aby nevznikaly chyby při překladu. Pokud to nechcete zahrnovat, lze tento krok obejít a všechny soubory z knihovny vložit do složky "src", případně, kde máte umístěn soubor "main.c". Po provedení tohoto postupu můžete začít pracovat s knihovnou.

6.3 Použití knihovny

- Naimportování souborů knihovny
- Pro použití vybraného řadiče je potřeba zahrnout potřebný soubor do vašeho "main"projektu.
  - Pro použití řadiče KS0108 zahrňte soubor #include "B_KS0108.h"
  - Pro použití řadiče ST7565 zahrňte soubor #include "B_ST7565.h"
  - Pro použití řadičů ST7066 zahrňte soubor #include "B_ST7066.h"
  - Pro použití řadičů HD44780 zahrňte soubor #include "B_HD44780.h"
  - Pro použití řadičů ST7735, IL9163 a GCC9A01 zahrňte soubor #include "B_ST7735.h"
  - Pro použití řadičů ST7789 zahrňte soubor #include "B_ST7789.h"
- Dalším krokem je výběr řadiče, vždy si lze vybrat jen jeden řadič ze všech.
- Pro všechny grafické řadiče jsou nadefinovány velikosti buffru v souboru Vram.h. Pokud budete využívat vlastní funkce pro grafiku je potřeba ho zahrnout v podobě #include "Vram.h"
- Aktivace vybraného řadiče, pomocí příkazu define, který můžete umístit přímo v souboru Vram.h, případně do mainu.c pokud jste provedly předchozí krok:
  - Výběr řadiče KS0108 přidejte příkaz #define KS0108

    - Výběr řadiče ST7565 přidejte příkaz #define ST7565
    - Výběr řadiče ST7735 přidejte příkaz #define ST7735
    - Výběr řadiče IL9163 přidejte příkaz #define IL9163
    - Výběr řadiče GC9A01 přidejte příkaz #define GC9A01
    - Výběr řadiče ST7789 přidejte příkaz #define ST7789
    - Výběr řadiče ST7066, není potřeba nic definovat.
    - Výběr řadiče HD44780, není potřeba nic definovat.

- Inicializaci řadičů proveď následovně:
    - Pro řadič KS0108 je inicializace Init_KS0108()
    - Pro řadič ST7565 je inicializace Init_ST7565()
    - Pro řadič ST7735 je inicializace Init_ST7735()
    - Pro řadič IL9163 je inicializace Init_ST7735()
    - Pro řadič GC9A01 je inicializace Init_ST7735()
    - Pro řadič ST7789 je inicializace Init_ST7789()
    - Pro řadič ST7066 je inicializace Init_ST7066()
    - Pro řadič HD44780 je inicializace Init_HD44780()
- V paměti RAM mikrokontroléru se vytvořilo pole LCD_VideoRam[] o patřičné velikosti dle použitého řadiče. Pole se nevytváří pro alfanumerické řadiče.
- Pro odeslání dat do řadičů:
    - Pro grafické řadiče jsou k dispozici funkce:
        - Pro odeslání dat do řadiče KS0108 VRAM_Send_KS0108()
        - Pro odeslání dat do řadiče ST7565 VRAM_Send_ST7565()
        - Pro odeslání dat do řadiče ST7735 VRAM_Send_ST7735()
        - Pro odeslání dat do řadiče IL9163 VRAM_Send_ST7735()
        - Pro odeslání dat do řadiče GC9A01 VRAM_Send_GC9A01()
        - Pro odeslání dat do řadiče ST7789 VRAM_Send_ST7789()
    - Pro alfanumerické řadiče jsou k dispozici funkce:
        - Pro odeslání znaků do řadiče ST7066 je zde funkce, kde parametrem funkce jsou znaky v uvozovkách. Send_String_ST7066()

- Pro odeslání znaků do řadiče HD44780 je zde funkce, kde parametrem funkce jsou znaky v uvozovkách. Send_String_HD44780()
- Nastavení pozice kurzoru pro řadič ST7066, parametr x a y představují pozici na displeji. Set_Cursor_ST7066(x, y)
- Nastavení pozice kurzoru pro řadič HD44780, parametr x a y představují pozici na displeji. Set_Cursor_HD44780(x, y)

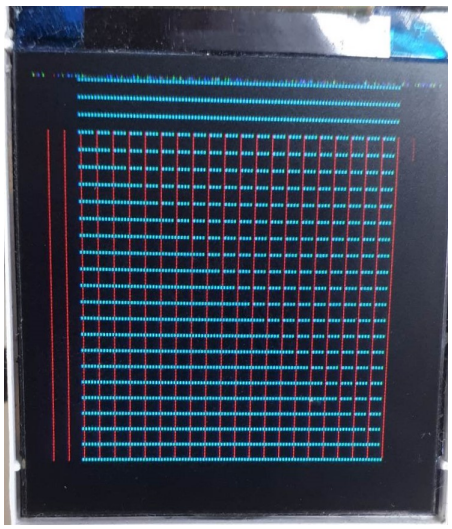
6.4 Ukázka použití knihovny

Pro přiblížení pochopení použití knihovny je ve výpisu kódu 6.1 zobrazena hlavní smyčka programu. Hlavní smyčka vykonává vykreslování vertikálních a horizontálních čar ve dvou různých barvách. Výsledek vyobrazení je na obrázku 6.2.

```
#include <stm32f4xx.h>

#include "stm_core.h"
#include "stm_systick.h"
#include "Grafic_16Color.h"
#include "B_ST7535.h"

int main(void) {
  InitSystickDefault();
  Init_ST7535();

  Draw_Color_VLine(5, 20, 15, YELLOW);
  Draw_Color_HLine(120, 0, 15, RED);
  Draw_Color_Line(120, 120, 110, 80, BLUE);
  Draw_Color_Elipse(50, 50, 10, 10, GREEN);
  Draw_Color_Elipse(15, 25, 20, 32, PURPLE);
  Draw_Color_Elipse(100, 90, 10, 25, BLUE);
  Draw_Color_Elipse(10, 90, 60, 35, CYAN);
  VRAM_Send_ST7535();

  WaitMs(2000);
  Draw_Color_Fill(0x0000);
  VRAM_Send_ST7535();

  uint16_t tm0 = 0, tm1 = 500;
  uint8_t h = 0, v = 0;
  while (1) {
```

```
28
29      if (tm0 < _ticks) {
30        tm0 = _ticks + 1000;
31        Draw_Color_HLine(h, 15, 100, RED);
32        h += 5;
33        VRAM_Send_ST7535();
34      }
35
36      if (tm1 < _ticks) {
37        tm1 = _ticks + 1000;
38        Draw_Color_VLine(15, v, 100, CYAN);
39        v += 5;
40        VRAM_Send_ST7535();
41      }
42
43      if (v >= 130 || h >= 130) {
44        Draw_Color_Fill(0x0000);
45        VRAM_Send_ST7535();
46        v = 0;
47        h = 0;
48      }
49    }
50 }
```

Výpis kódu 6.1 Ukázkové použití knihovny s řadičem IL9163



Obr. 6.2 Výsledné zobrazení s řadičem IL9163

7 Zhodnocení a závěr

7.1 Prozkoumání řadičů

Zadáním práce je prozkoumání řadičů pro malé LCD zobrazovače. V úvodní části 2 je vyobrazeno blokové schéma řadiče a popsány jednotlivé bloky v řadiči. Dále je popsána základní funkce řadiče, která spočívá v zobrazování dat na displeji. V případě nahrazení řadiče procesorem by došlo k přehlcení procesoru instrukcemi, které vykonává pro LCD zobrazovač a docházelo by k omezování hlavního programu procesoru.

7.2 Porovnání vlastností

Jedním z úkolů bakalářské práce je porovnání vlastností jednotlivých řadičů s osazenými zobrazovači. Konkrétně je v zadání uvedeno porovnat podporované rozlišení, barevnou hloubku, komunikaci, napájení a případně další parametry charakterizující jednotlivé zobrazovače. Všechny tyto parametry případně i další parametry jako třeba velikost úhlopříčky jsem uvedl do tabulky 7.1.

Komunikace s řadiči je realizována za pomocí sběrnic. Nejčastěji se zde opakuje sériová sběrnice SPI a 8 bitová paralelní sběrnice. Tyto sběrnice jsou hojně používané. Práce s těmito sběrnicemi je rozdílná a každá přináší jiné výhody i nevýhody. Z hlediska komfortu práce z jednotlivými sběrnicemi je sériová sběrnice SPI příznivější. Především z hlediska nižší náročnosti na počet vodičů a možnosti využití DMA.

Všechny testované barevné řadiče mají stejnou barevnou hloubku. Každý z nich dovoluje konfiguraci barevné hloubky v podobě nastavení kolik bitů bude představovat jeden pixel při komunikaci. Ovšem vnitřně dochází k dekódování barvy vždy na 18 bitový formát.

Každý řadič, který jsem měl k dispozici je připojen k displejům o různé velikosti úhlopříčky a různým počtem pixelů. Například při porovnání řadičů IL9163 a ST7789 je v tabulce vidět, jejich podobné velikosti úhlopříčky. Ale řadič ST7789 má dvojnásobnou hustotu pixelů na jeden cm^2.

Z hlediska spotřeby jednotlivých řadičů je v tabulce 7.1 vidět, že všechny řadiče bez připojené zátěže mají spotřebu do 1 mA. Ovšem některé vyobrazené řadiče mají napájecí napětí 5 V, to může být lehce matoucí. Jelikož všechny řadiče pracují interně na 3,3 V případně i na nižších napěťových úrovních. Pro snížení výstupního napětí se využívají LDO stabilizátory napětí. Který přebytečnou energii přivedenou na vstup přemění na teplo, množství vzniklého ztrátového tepla lze dopočítat pomocí rovnice 2.1. Velikosti proudů tekoucí do stabilizátoru a z něj jsou velice podobné, proud spotřebovaný na stabilizátoru je velice malí, vzhledem k celkové velikosti odebíraného proudu. Řádově se pohybuje kolem desítek uA. Proto jsem velikost tohoto proudu zanedbal. Spotřeba je nejvíce ovlivněna jakmile se k řadiči připojí zátěž včetně podsvícení displeje. Nejnáročnějším

Tab. 7.1 Porovnání vlastností jednotlivých řadičů

Název řadiče	Komunikace	Barevná hloubka	Úhlopříčka		Rozlišení pixelů		Napájecí napětí [V]	Odebíraný proud [mA]
			in	cm	šířka	výška		
KS0108	8 bit paralerní	1	2,9	7,36	128	64	5	0,5
ST7565	SPI	1	1,57	4,00	128	32	3,3	0,2
ST7066	IIC 8 bit paralérní	1	2,63	6,7	80	16	5	0,4
HD44780	IIC 8 bit paralérní	1	3,14	8,00	100	32	5	0,4
ST7735	SPI	262 144	1,80	4,57	128	120	3,3	0,5
IL9163	SPI	262 144	1,44	3,65	128	128	3,3	1,0
ST7789	SPI	262 144	1,30	3,30	240	240	3,3	1,0
GC9A01	SPI	262 144	1,28	3,25	240	240	3,3	1,0

prvkem na spotřebu je podsvícení zobrazovače, u malých zobrazovačů udělá rozdíl v odebíraném proudu až desítky mA.

7.3 Zhodnocení knihovny

Zhodnocení mého plánu vytvořit knihovnu, která bude efektivně pracovat s LCD řadiči a urychlí práci s nimi. Ve vytvořené knihovně je implementováno několik řadičů konkrétně pro černobílé zobrazovače se jedná o řadiče KS0108, ST7565, ST7066 a HD44780. A pro barevné LCD obrazovky jde o řadiče ST7735, IL9163, GC9A01 a ST7789. Knihovna je funkční a v kapitole 6 je uveden návod na zprovoznění knihovny. Knihovna zprostředkovává komunikaci mezi řadičem mikrokontrolérem a vytváří buffer v RAM paměti mikrokontroléru. Pro zápis do bufferu, u monochromatických zobrazovačů jsou funkce poměrně rozsáhlé dovolují vykreslovat jak text tak i grafiku v podobě čar, kruhů, elips a dalších obrazců. Oproti tomu funkce pro zápis do bufferu určené pro barevné zobrazovače nejsou tak rozsáhlé, ale zvládají vykreslovat čáry, elipsy a kruhy. Rozhodně by stálo je ještě zdokonalit.

Knihovna rozhodně není naprogramována nejefektivněji a najdou se části co potřebují zdokonalit. Rozhodně stojí za zlepšení a zjednodušení výběru konkrétního řadiče, aby se nemuselo vybírat v souboru Vram.h, ale provádět tento výběr rovnou v hlavní smyčce programu. Rozhodně dalším zlepšením je přidáním nahrávání obrazových dat do řadičů za pomocí DMA.

Literatura

[1] Texas Instruments Incorporated (TI). *PCF8574 Remote 8-Bit I/O Expander for I 2C Bus*. online. URL: https://www.ti.com/lit/ds/symlink/pcf8574.pdf.

[2] Siltronic AG. *ST7066U - Dot Matrix LCD Controller/Driver*. online. URL: https://newhavendisplay.com/content/app_notes/ST7066U.pdf.

[3] Siltronic AG. *ST7735 - 262K Color Single-Chip TFT Controller/Driver*. online. URL: https://www.displayfuture.com/Display/datasheet/controller/ST7735.pdf.

[4] carlfriess. *GC9A01 Demo*. online. URL: https://github.com/carlfriess/GC9A01_demo/blob/main/GC9A01.c.

[5] Cburnett. *SPI three slaves*. online. URL: https://commons.wikimedia.org/wiki/File:SPI_three_slaves.svg.

[6] Cburnett. *SPI three slaves*. online. URL: https://upload.wikimedia.org/wikipedia/commons/0/04/I2C_controller-target.svg.

[7] ILI TECHNOLOGY CORP. *TFT LCD Single Chip Driver 132RGBx162 Resolution and 262K color*. online. URL: https://www.hpinfotech.ro/ILI9163.pdf.

[8] Seiko Epson Corporation. *S1D15E06 Series*. online. URL: https://www.rockbox.org/wiki/pub/Main/DataSheets/epson_s1d15e06.pdf.

[9] the free encyclopedia From Wikipedia. *I^2C*. online. URL: https://en.wikipedia.org/wiki/I%C2%B2C.

[10] the free encyclopedia From Wikipedia. *Serial Peripheral Interface*. online. URL: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface.

[11] Ltd. Hitachi. *HD44780U (LCD-II)*. online. URL: https://www.sparkfun.com/datasheets/LCD/HD44780.pdf.

[12] GalaxyCore Inc. *GC9A01A - a-Si TFT LCD Single Chip Driver 240RGBx240 Resolution*. online. URL: https://www.buydisplay.com/download/ic/GC9A01A.pdf.

[13] CSc. prof. Ing. Jiří Pinker. „Číslicové elektronické systémy“.

[14] Arm mBed. *mbed Application Shield*. online. URL: https://os.mbed.com/components/mbed-Application-Shield/.

[15] Ltd. Samsung Electronics Co. *KS0108B - 64CH SEGMENT DRIVER FOR DOT MATRIX LCD*. online. URL: https://www.sparkfun.com/datasheets/LCD/ks0108b.pdf.

[16] STMicroelectronics. *STM32 Nucleo-64 development board with STM32F411RE MCU*. online. URL: https://www.st.com/en/evaluation-tools/nucleo-f411re.html.

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Seznam příloh

Příloha A Obsah přiloženého archivu

Obsahem přiloženého archivu jsou všechny kódy a jejich součásti ve finální podobě. V obrázku A.1 je znázorněno uspořádání přílohy v archivu.

```
Grafic_16Color ..... Grafika pro barevné zobrazovače
├── Grafic_16Color.h
└── Grafic_16Color.c
Grafic_Mono ..... Grafika pro monochromatické zobrazovače
├── font_8x8.h
├── font_atari_8x8.h
├── font_atari_8x8.c
├── font_thin_8x8.h
├── font_thin_8x8.c
├── font_type.h
├── Grafic_Mono.h
└── Grafic_Mono.c
I2C
├── nucleo_i2c.h ..... Hlavičkový soubor pro konfiguraci sběrnice I2C
└── nucleo_i2c.c ..... Hlavní kód pro sěrnici I2C
LCD_Knihovna
├── KS108
│   ├── B_KS108.h ..... Hlavičkový soubor pro řadič KS0108
│   ├── B_KS108_Test.c ..... Hotové ukázkové funkce
│   └── B_KS108.c ..... Hlavní kód řadiče
├── ST7066
│   ├── B_ST7066.h ..... Hlavičkový soubor pro řadič ST7066
│   └── B_ST7066.c ..... Hlavní kód řadiče
├── HD44780
│   ├── B_HD44780.h ..... Hlavičkový soubor pro řadič HD44780
│   └── B_HD44780.c ..... Hlavní kód řadiče
├── ST7535
│   ├── B_ST7535.h ..... Hlavičkový soubor pro řadičů ST7535, IL9163 a GCCA01
│   └── B_ST7535.c ..... Hlavní kód řadičů
├── ST7789
│   ├── B_ST7789.h ..... Hlavičkový soubor pro řadič ST7789
│   └── B_ST7789.c ..... Hlavní kód řadiče
├── ST7565
│   ├── B_ST7565.h ..... Hlavičkový soubor pro řadič ST7565
│   └── B_ST7565.c ..... Hlavní kód řadiče
├── Vram
│   └── Vram.h ..... Nastavení velikosti bufru
└── ReadMe.txt ..... Informace pro práci s knihovnou
PMK_Shield
├── mbed_Shield.h ..... Hlavičkový soubor pro konfiguraci Mbed Shieldu
└── mbed_Shield.c ..... Hlavní kód pro Mbed Shield
SPI
├── B_SPI.h ..... Hlavičkový soubor pro konfiguraci sběrnice SPI
└── B_SPI.c ..... Hlavní kód pro sěrnici SPI
STM_Core
├── nucleo_usart.h
├── nucleo_usart.c
├── stm_core.h
├── stm_core.c
├── stm_systick.h
└── stm_systick.c
Universal
└── B_Define_Core.h
```

Obr. A.1 Uspořádání souborů v příloze