VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

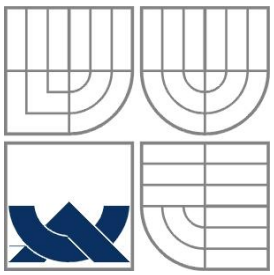
GRAFICKÁ KNIHOVNA PRO DOTYKOVÝ LCD

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

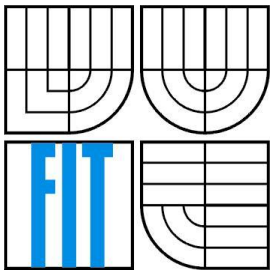
AUTOR PRÁCE
AUTHOR

TOMÁŠ NOŽIČKA

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GRAFICKÁ KNIHOVNA PRO DOTYKOVÝ LCD

GRAPHICS LIBRARY FOR TOUCHSCREEN LCD

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ NOŽIČKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PAVEL BARTOŠ

BRNO 2011

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2010/2011

Zadání bakalářské práce

Řešitel: **Nožička Tomáš**
Obor: Informační technologie
Téma: **Grafická knihovna pro dotykový LCD**
Graphics Library for Touchscreen LCD
Kategorie: Vestavěné systémy

Pokyny:

1. Seznamte se s výukovým kitem, který obsahuje mikrokontrolér a barevný dotykový LCD.
2. Seznamte se s komunikačním protokolem mezi řídicím mikrokontrolérem a displejem.
3. Navrhněte a implementujte ovladač displeje (inicializace, smazání, rozsvícení bodu ...).
4. Navrhněte a implementujte grafickou knihovnu pro displej, která bude obsahovat běžné ovládají prvky a zobrazovače dat (tlačítko, posuvník, text, výběrové pole apod.).
5. Vytvořte jednoduchou aplikaci, která prvky z knihovny bude vhodně demonstrovat.
6. Zhodnoťte dosažené výsledky a porovnejte je s jinými grafickými knihovnami.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bartoš Pavel, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2010
Datum odevzdání: 18. května 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Tato práce se zabývá grafickou knihovnu pro přídavný modul s dotykovým LCD k výukovému kitu s MCU MC9S08JM60 od firmy Freescale. Práce popisuje použitý mikrokotrolér, různé varianty komunikace s řadičem displeje a technologie dotykových vrstev displejů. Zbývající část práce se zabývá rozdělením knihovny na vrstvy, jejich popisu a způsobům použití.

Abstract

This work deals with graphics library for LCD module with touchscreen which is mounted on educational kit with Freescale MCU MC9S08JM60. The thesis describes used microcontroller, different ways of communication witch LCD driver and touchscreen technologies. Remaining part of this thesis deals with library layers, their description and ways to use it.

Klíčová slova

grafická knihovna, dotykový displej, MCU, Freescale, LCD, TFT, MC9S08JM60, HCS08, vestavěný systém

Keywords

graphics library, touchscreen, MCU, Freescale, LCD, TFT, MC9S08JM60, HCS08, embedded system

Citace

Nožička Tomáš: Grafická knihovna pro dotykový LCD, bakalářská práce, Brno, FIT VUT v Brně, 2011

Grafická knihovna pro dotykový LCD

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Pavla Bartoše.
Další informace mi poskytl Ing. Václav Šimek.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Tomáš Nožička
18. května 2011

Poděkování

Tímto děkuji Ing. Pavlu Bartošovi a Ing. Václavu Šimkovi za pomoc a cenné rady, které mi poskytli při vývoji této bakalářské práce.

Obsah

1 Úvod

S dotykovými displeji se v posledních letech setkáváme téměř každodenně. Jejich největší podíl je zastoupen mobilními telefony, kde se jejich přítomnost ve střední a vyšší třídě stává standardem. Mají ale daleko širší škálu použití sahající od různých průmyslových systémů až po tablety. Přinášejí příjemnější a mnohdy i rychlejší uživatelské rozhraní tam, kde není zapotřebí rozsáhlých textových vstupů od uživatele.

Cílem mé práce bylo vyvinout grafickou knihovnu pro dotykový LCD (Liquid Crystal Display). To zahrnuje vytvoření ovladače displeje, ovladače dotykové vrstvy a grafické knihovny obsahující základní funkce pro kreslení tvarů (čára, obdélník atd.) a objektovou nadstavbu (tlačítka, textové pole ad.).

V první kapitole se věnuji výukovému kitu. Je zde popsáno jeho vybavení, možnosti napájení a programování. Dále se zde nachází podrobnější popis osazeného mikrokontroléru a také dotykového displeje.

Druhá kapitola se zabývá displejem a jeho dotykovou vrstvou. Jsou zde popsány piny k jeho ovládání. Následuje popis komunikačních rozhraní, které nám ovladač displeje nabízí. Dále jsou zde popsány různé technologie dotykových vrstev.

Celá třetí kapitola je věnována implementaci. Zde lze nalézt rozdělení knihovny a popis jednotlivých vrstev. Na konci kapitoly jsou zmíněny nejdůležitější algoritmy, které knihovna implementuje.

Čtvrtá kapitola popisuje způsob práce s knihovnou a nutné nastavení před prvním použitím. Je zde také popsáno, které moduly mikrokontroléru knihovna využívá, aby se předešlo případným kolizím.

V závěru je zhodnocena knihovna v porovnání s ostatními. Také jsou zde popsány možná rozšíření knihovny do budoucna.

2 O KITu

Hardwarový kit, který je určen k výuce v předmětu IMP, je vybaven mikrokontrolérem (dále MCU – Microcontroller Unit), dvěma kapacitními klávesnicemi, čtečkou SD karet, USB a VGA konektorem, segmentovým displejem, směrovými tlačítky, potenciometrem a přídavným modulem s dotykovým LCD. Pod tímto modulem se nachází i integrovaný programátor založený na čipu JS16, který se k PC připojuje pomocí USB konektoru, z něhož je i napájen.

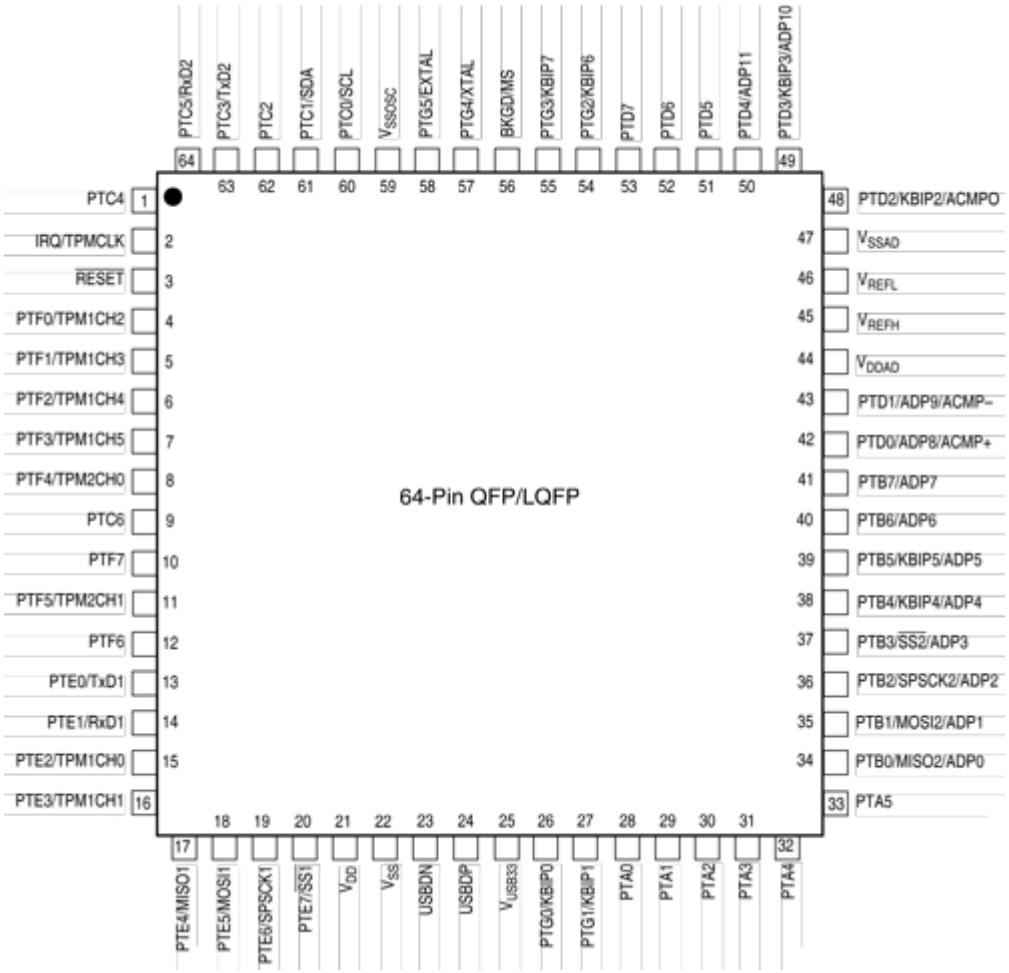
Obrázek 2.1: Výukový kit

Kit lze napájet pomocí USB konektoru, externího adaptéru nebo externího programátoru, který se připojí přes konektor „P7“.

2.1 Mikrokontrolér

Kit je osazen MCU MC9S08JM60 od firmy Freescale. Ten pochází z řady HCS08, která vyniká sníženou spotřebou. Jako zdroj informací o MCU byl použit datasheet [1] a kniha [2].

2.1.1 Pouzdro



Obrázek 2.2: MCU na kitu (převzato z [1])

MCU na kitu je osazeno v 64pinovém pouzdře, které zajišťuje dostatečný počet pinů k ovládání všech přítomných periferií. Pouzdro je typu LQFP (Low-profile Quad Flat Package), což znamená, že piny jsou rozmístěny rovnoměrně ke všem čtyřem stranám a pouzdro je snížené oproti QFP (Quad Flat Package).

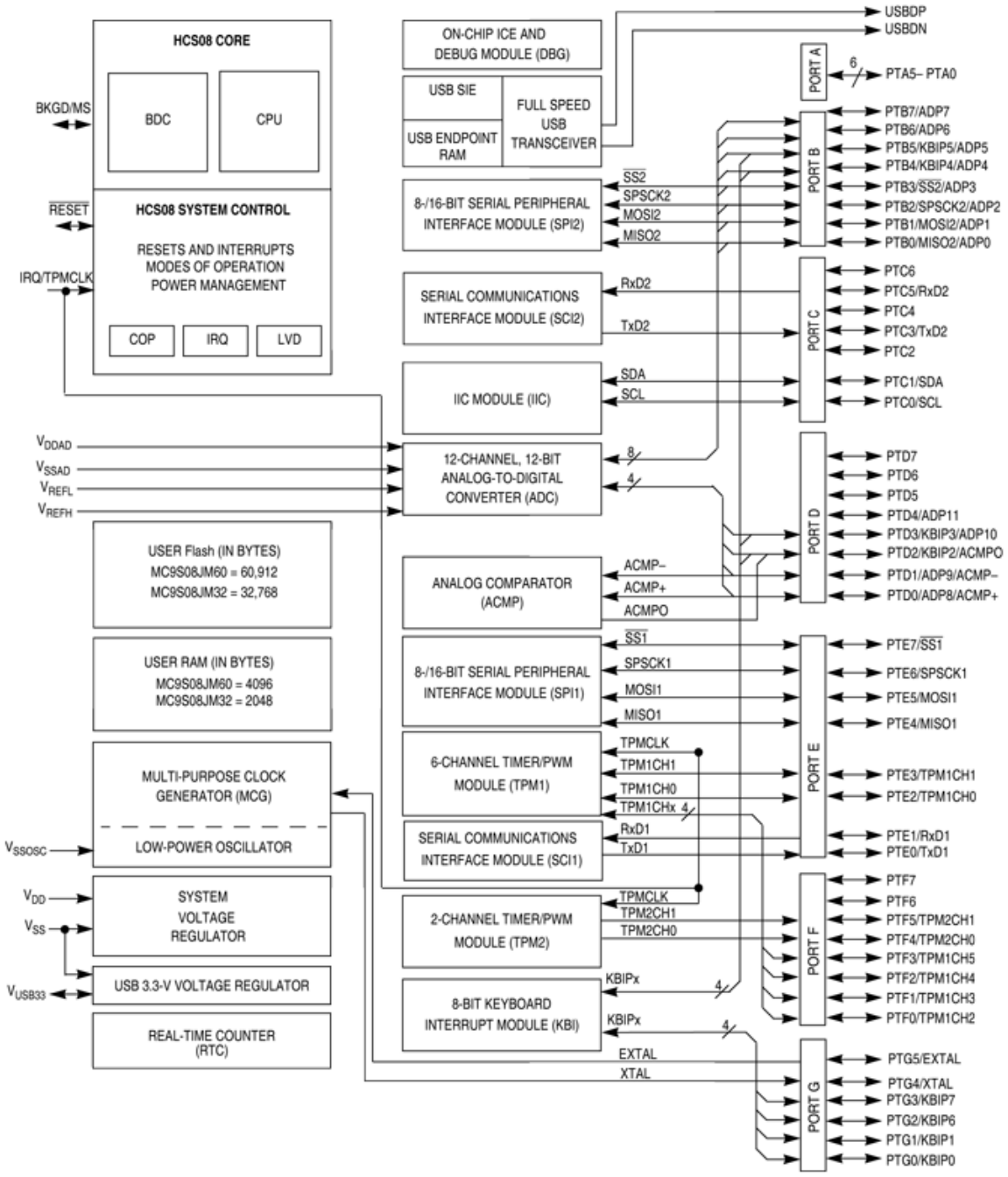
2.1.2 Vybavení

MCU je vybaven 4 096 B RAM, 256 B USB RAM a 60 912 B flash paměti, která slouží pro uložení programu. Z periferií pak obsahuje:

- ACMP (Analog Comparator Module)
- 12kanálový ADC (Analog to Digital Converter)
- I2C (Inter-Integrated Circuit)
- IRQ (Interrupt ReQuest) modul
- 8pinový KBI (Keyboard Interrupt) modul
- SCI1 a SCI2 (SCI – Serial Communications Interface)
- SPI1 a SPI2 (SPI – Serial Peripheral Interface)

- 6kanálový TPM1 a 2kanálový TPM2 (TPM – Timer/Pulse-Width Modulator)
- USB (Universal Serial Bus) kontrolér specifikace 2.0 (vyžaduje frekvenci sběrnice na 24 MHz a referenční signál o 48 MHz)
- 51 vstupně/výstupních pinů

Jejich zapojení v MCU znázorňuje obrázek 2.3.



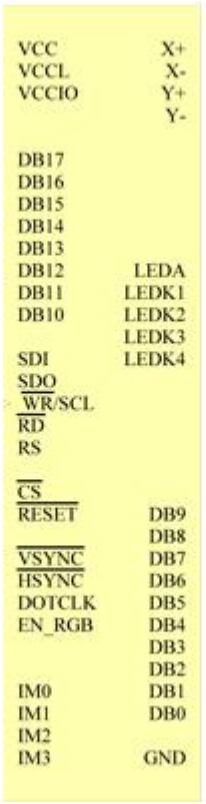
Obrázek 2.3: Zapojení periferií v MCU (převzato z [1])

3 Displej

Přídavný modul je osazen displejem DT028TFT-TS od firmy Displaytech. Jako zdroj informací pro tuto kapitolu byl použit datasheet [4]. Displej má rozlišení 240 na 320 pixelů, 262 144 barev, jeho úhlopříčka činí 2,8" a k jeho řízení slouží zabudovaný driver ILI9320 (dále jen driver). Displej je vyroben technologií TFT (Thin Film Transistor) a k jeho podsvícení jsou použity čtyři bílé LED (Light-Emitting Diode), které jsou napájeny integrovaným obvodem LDS8845. Driver obsahuje 172 800 bitů grafické paměti, což odpovídá danému rozlišení, v němž je každá barva uložena na 6 bitech. Poskytuje také např. možnost přepnout na zobrazení v osmi barvách nebo přechod do režimu spánku. Díky jeho možnostem softwarově přesně řídit spotřebu, je dobrou volbou do přenosných zařízení, jako jsou mobilní telefony.

3.1 Popis pinů

Úplný popis všech pinů by zde postrádal smysl, proto se budu věnovat jen těm, které knihovna používá. (Komunikace je řešena pomocí systémového rozhraní – více v 3.2.1.)



Obrázek 3.1: Schéma DT028TFT-TS (převzato z [5])

Při komunikaci pomocí systémového rozhraní i80 jsou používány následující piny:

- IM0-IM3: jejich kombinací se nastavuje typ komunikačního rozhraní
- nCS: „Chip select" – povoluje komunikaci s čipem displeje
- RS: „Register select" – rozlišuje přístup k registrům a datům
- nWR: při nízké hladině povoluje operaci zapsání dat
- nRD: při nízké úrovni povoluje operaci čtení dat

- DB[17:0]: 18bitová paralelní obousměrná datová sběrnice

3.2 Komunikace s displejem

Driver obsahuje následující čtyři typy komunikačních rozhraní: i80, VSYNC, SPI a RGB. Jejich principy budou zmíněny v následujících podkapitolách.

3.2.1 Systémové rozhraní i80/8 bitů

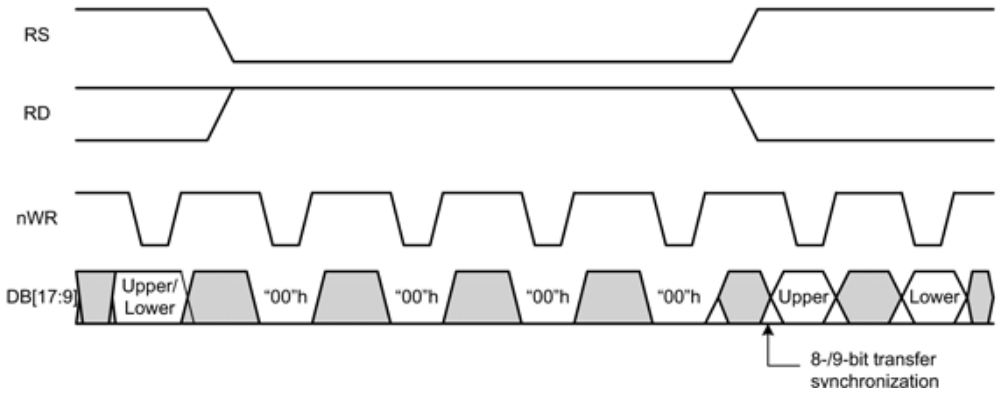
Rozhraní i80/8 bitů se vybírá nastavením IM[3:0] na hodnotu „0011“ a k paralelnímu datovému přenosu jsou použity piny DB17-DB10 (na kitu jsou mapovány na port B). Rozhraní i80 obsahuje také varianty i80/18, 16 a 9 bitů, jejichž princip je podobný, a potřebné informace lze najít v [4]. Na kitu jsou však datové vývody DB9-DB0 uzemněny a nelze je tedy použít. Navíc je kit v současné konfiguraci osazen pouze 8bitovým MCU. Dále se tedy budeme věnovat pouze 8bitové verzi tohoto rozhraní.

Při komunikaci přes toto rozhraní se budeme snažit komunikovat se třemi registry: 16bitovým IR (Index Register), 18bitovým WDR (Write Data Register), 18bitovým RDR(Read Data Register). Jejich použití pro komunikaci s driverem ukazuje tabulka 3.1.

Funkce	RS	nWR	RD
Zápis indexu do IR	0	0	1
Zápis do řídícího registru nebo GRAM pomocí WDR	1	0	1
Čtení z řídícího registru nebo GRAM pomocí RDR	1	1	0

Tabulka 3.1: Komunikace s driverem

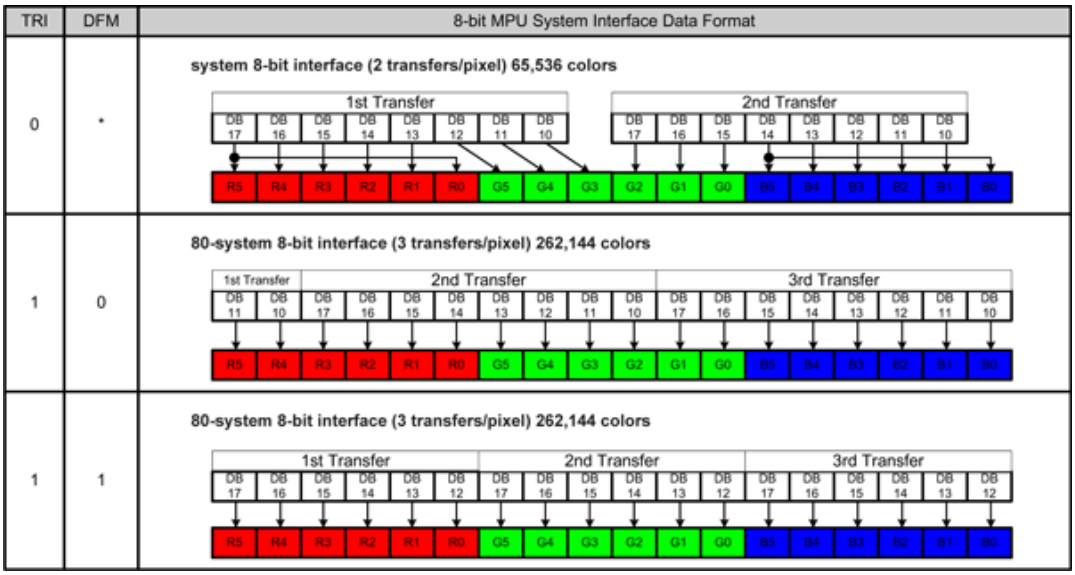
Protože IR je 16bitový, musíme data po sběrnici poslat 2krát po 8 bitech, kde je jako první poslána horní (významnější) polovina. Přenos zobrazuje obrázek 3.2, spolu se synchronizací (přenos 4krát „00h“). Ta však není povinná, ale je vhodné jí jednou za čas provést a předcházet tak chybám.



Obrázek 3.2: Časový průběh přenosu (převzato z [4])

Vzhledem k tomu, že v GRAM displeje je každá barva uložena na 6 bitech, je nutné přenést 18 bitů po 8bitové sběrnici. Z toho vyplývá, že je třeba uskutečnit 3 datové přenosy. Tato možnost je zobrazena na obrázku 3.3 ve dvou obdobných variantách. Driver však poskytuje i třetí možnost, která potřebuje přenosy jen dva. To představuje redukci přenosu o 50 %. Zde je nutné si uvědomit, že

reálná aplikace bude muset obnovovat např. pozadí, tedy všech 76 800 bodů. A zde již tato optimalizace přináší výrazné zrychlení. Nutnou cenou za ní je ztráta posledního bitu u červené a modré barvy. Tyto dvě barvy byly vybrány výrobcem záměrně, protože je na ně naše oko méně citlivé než na barvu zelenou. V tomto případě tedy máme k dispozici z původních 262 144 barev jen 65 536. I to je však dostatečné a za úsporu procesorového času MCU to stojí. Navíc takto přijdeme jen o ty nejjemnější odstíny, které mají ve výsledném obraze jen nepatrný vliv.



Obrázek 3.3: Přenos barevných složek (převzato z [4])

3.2.2 Rozhraní SPI (Serial Peripheral Interface)

Rozhraní SPI se volí nastavením hodnoty „010x“ na pinech IM[3:0]. Jak již z názvu vyplývá, jedná se o sériový přenos. Rozhraní se aktivuje při sestupné hraně na pinu nCS a ukončí se nástupnou hranou na tomto pinu. Na začátku komunikace je do driveru přenesen „start byte“, jehož formát znázorňuje tabulka 3.2. Ten obsahuje adresu zařízení (lze měnit pouze její poslední bit nastavením IM0) a zároveň i specifikaci požadované operace, jak znázorňuje tabulka 3.3.

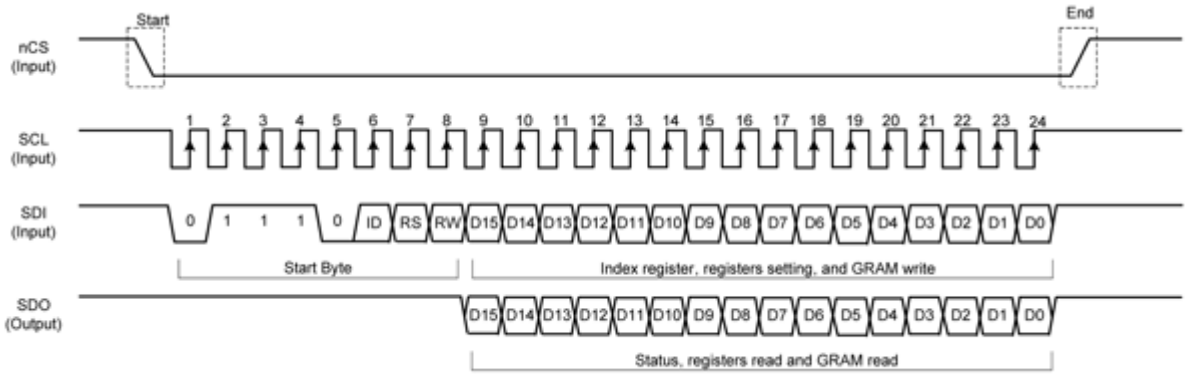
Přenášené bity	Start bit	1	2	3	4	5	6	7	8
Formát	Začátek přenosu	Identifikace zařízení						RS	R/W
Hodnoty		0	1	1	1	0	ID/IM0	1/0	1/0

Tabulka 3.2: Formát „start bytu“

RS	R/W	Operace
0	0	Nastaví IR
0	1	Přečte status
1	0	Zapíše do registru nebo GRAM
1	1	Přečte registr nebo GRAM

Tabulka 3.3: Význam kombinací ve „start bytu“

Poté, co driver dekóduje „start byte“, je přenos zahájen. Protože je každý přenos 8bitový, tak pro přístup k registrům musí být proveden 2krát. U přenosu dat do GRAM je počet barev automaticky snížen z 262 144 na 65 536 způsobem popsaným v kapitole 3.2.1.

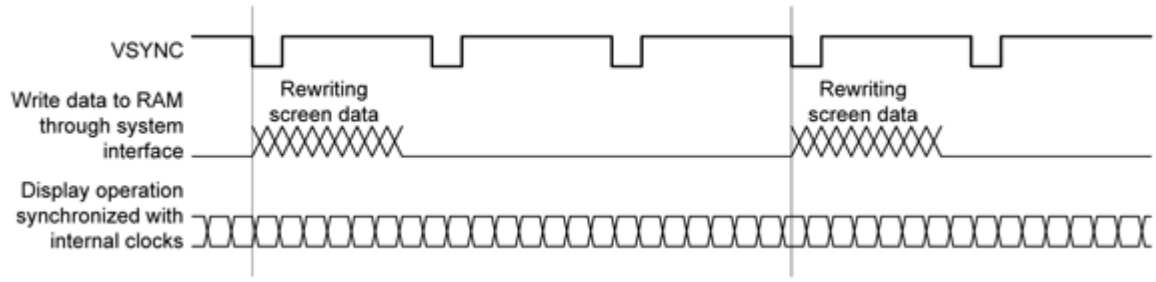


Obrázek 3.4: Základní přenos přes SPI (převzato z [4])

Díky tomu, že je přenos sériový, je oproti paralelnímu přenosu přes rozhraní i80 výrazně pomalejší a stává se tak jen těžko použitelným pro aplikaci typu grafická knihovna. Navíc při většině přenosů je nutno několikrát načíst neplatná data než jsou k dispozici data platná (blíže popsáno v [4]). Jeho výhodou je to, že protokol je jednoduchý a přenos je obstaráván SPI modulem umístěným v MCU, který tak přebírá část naší práce.

3.2.3 Rozhraní VSYNC

Toto rozhraní je rozšířením systémového rozhraní i80 o signál VSYNC a umožňuje nám využít funkci pohyblivého obrazu. Pokud je rozhraní s touto funkcí zvoleno, je automaticky stanovena minimální frekvence obnovy GRAM na 5,7 MHz. Rozhraní je povoleno nastavením DM[1:0] na hodnotu „10“ a RM na „0“, kde DM[1:0] a RM jsou pojmenované bity registru driveru číslo $0C_{16}$ (RGB Display Interface Control 1) dle [4].

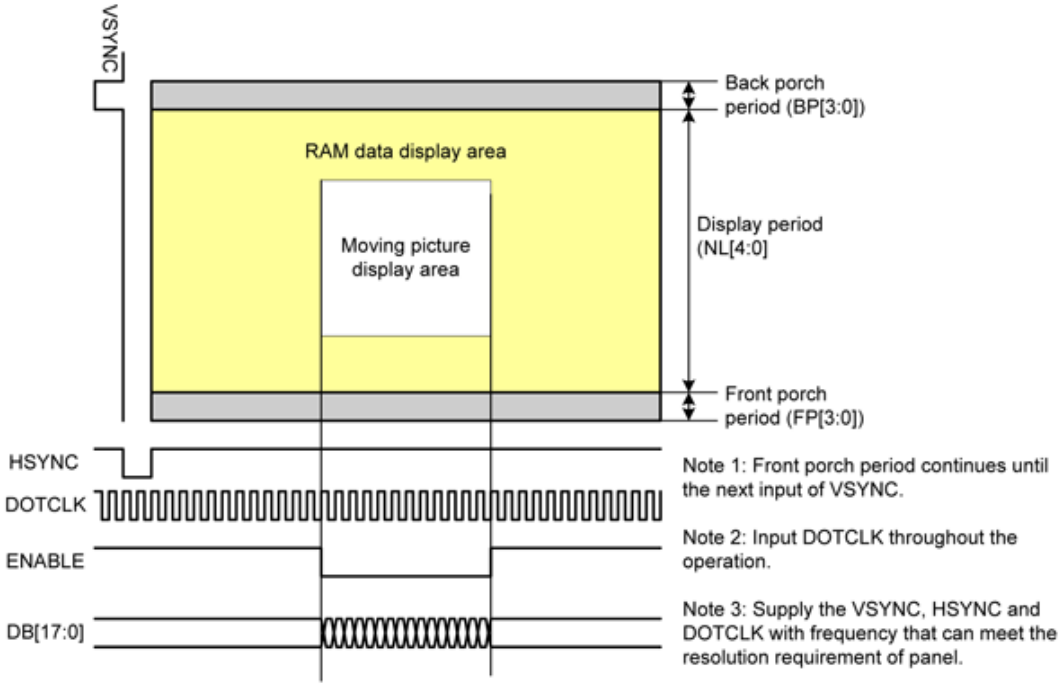


Obrázek 3.5: Přenos přes rozhraní VSYNC (převzato z [4])

Při použití tohoto rozhraní je nutné dodržet minimální frekvenci obnovy GRAM. Protože je perioda obnovy jedné obrazovky displeje určena signálem VSYNC, je nutné, aby byla delší než vnitřní perioda obnovy celého displeje. Pokročilejší funkce, jako je vertikální scrollování, nejsou v tomto režimu podporovány.

3.2.4 Rozhraní RGB

Komunikace s displejem je v tomto rozhraní řízena signály VSYNC, HSYNC a DOTCLK. Je podporován přenos se šířkou sběrnice 18, 16 a 6 bitů a data jsou ukládána do GRAM. Volba používaného typu se provádí nastavením bitů RIM[1:0] registru číslo $0C_{16}$ (RGB Display Interface Control 1) na příslušné hodnoty [4]. Přenos dat přes toto rozhraní je znázorněn na obrázku 3.6. Podrobnější popis a detailnější nákresy časování lze nalézt taktéž v [4].



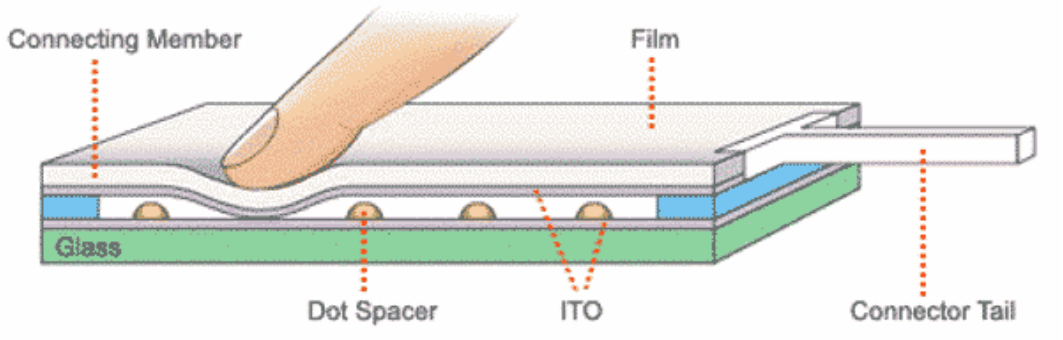
Obrázek 3.6: Přenos dat do GRAM přes rozhraní RGB (převzato z [4])

3.3 Touchscreen

3.3.1 Technologie dotykových displejů

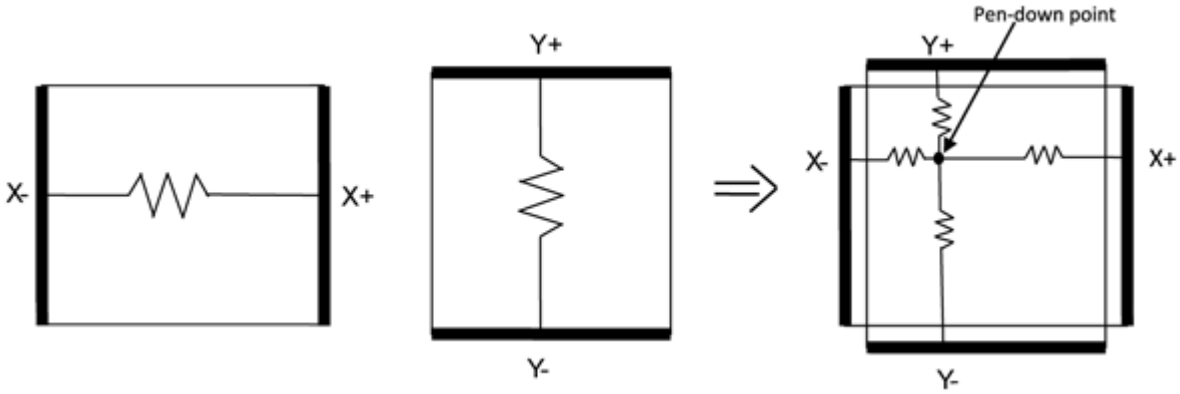
3.3.1.1 Rezistivní technologie

Následující informace byly získány z [6]. Rezistivní technologie se v principu skládá ze dvou vodivých průhledných vrstev, které jsou od sebe odděleny a stiskem prstu (či dotykem nějakého předmětu – např. stylusu) se spojí – viz. obrázek 3.7. Vzniká tak napěťový dělič, ze kterého můžeme odečíst napětí.



Obrázek 3.7: Provedení rezistivního dotykového displeje (převzato z [7])

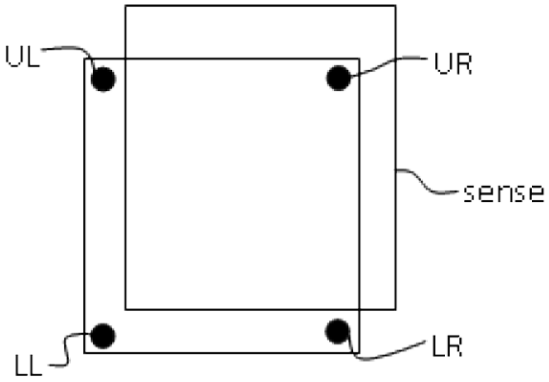
1) 4drátové dotykové rozhraní



Obrázek 3.8: Schéma 4drátového dotykového displeje (převzato z [6])

Konstrukce se skládá ze dvou průhledných a vodivých desek, na jejichž protějších koncích jsou vyvedeny kontakty. Tyto desky jsou oproti sobě otočeny o 90°. Zaměřme se nyní na obrázek 3.8. Pokud připojíme na pin Y+ napájecí napětí, Y- uzemníme a X- i X+ necháme odpojeny, vzniká nám při stisku napěťový dělič. V praxi pak tedy změříme napětí pomocí A/D převodníku v MCU na pinu X+ nebo X-. Pokud v tomto postupu prohodíme piny X a Y, změříme tak napětí i na druhé ose. Protože odpor mezi Y- a Y+ (samozřejmě také i mezi X- a X+) je lineární, platí, že poměr napětí na odporovém děliči je stejný jako poměr vzdáleností k okrajům. Praktický výpočet s rovnicí následuje v kapitole 3.3.2.

2) 5drátové dotykové rozhraní



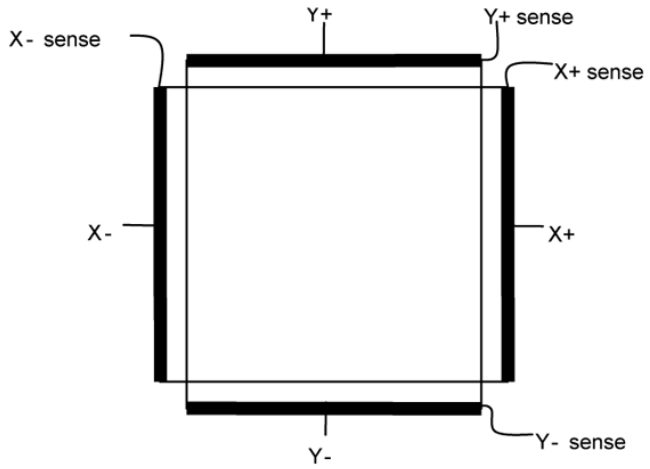
Obrázek 3.9: Schéma 5drátového dotykového displeje (převzato z [6])

U předchozího zapojení může nastat, že vrstvy vlivem opotřebení zůstanou v některém místě spojeny (vrchní zůstane promáčknuta) a výrazně to ovlivní měření A/D převodníku. Dle [6] je toto zapojení méně náchylné k této poruše.

Měření se provádí přivedením kladného napětí na UL i UR a opačnou polaritu na LL i LR. Pro měření druhé osy přivedeme napájecí napětí na UL i LL a opačnou polaritu na UR i LR. V obou případech měříme napětí na vodiči sense.

3) 8drátové dotykové rozhraní

Ve vodičích, spojích a neaktivních oblastech dotykového displeje vznikají vnitřní ztráty, jejichž důsledkem jsou chybně naměřené hodnoty. Proto je v tomto zapojení v druhé vrstvě pod každým kontaktem také čtecí kontakt, jak znázorňuje obrázek 3.10. A/D převod pak probíhá na obou pinech Y+ i Y-. Podobně i pro druhou osu.



Obrázek 3.10: 8drátové rozhraní dotykového displeje (převzato z [6])

3.3.1.2 Kapacitní technologie

Principem této technologie je tenká fólie (skládá se z několika vrstev), která je umístěna na displeji. Její vnitřní uspořádání vytváří kondenzátor, který je napájen čtyřmi elektrodami umístěnými v rozích displeje. Pokud se displeje dotkneme vodivým materiálem, vzniká tak na displeji parazitní kapacita, na kterou je z displeje přenesena část náboje. To se projeví zvýšením proudu, který je

dodáván elektrodami v rozích. Velikost proudu odebíraného z každé elektrody je úměrná její vzdálenosti k bodu dotyku.

Z toho plyne jedna z hlavních vlastností těchto displejů – k jejich ovládání je nutné vodivého materiálu (i lidská kůže je vodivá). Nelze jej tedy ovládat například v běžných zimních rukavicích. Tato technologie také přináší možnost detekce více dotyků najednou, na čemž jsou založena moderní uživatelská rozhraní především v mobilních zařízeních. Takto vyrobené displeje jsou více odolné, vyznačují se vysokou světelnou propustností a nevadí jim prach či mastnota.

Informace o této technologii byly získány z [8] a [9].

3.3.1.3 Technologie používající akustickou vlnu

Realizace této technologie je poměrně náročná a z technologií uvedených v kapitole 3.3.1 je nejdražší. Základem této technologie jsou vysílače ultrazvukového signálu ve dvou přilehlých stěnách a jeho přijímače na stranách protějších. Pokud je do takto vysílaných vln vložen předmět, je jím část vysílaných vln pohlcena. [9]

Výhodou této technologie je vysoké rozlišení, vysoká průzračnost a díky absenci přidaných vrstev vysoká mechanická odolnost. Nevýhodou je, že takovéto displeje musí být ovládány prstem nebo koženým ukazovátkem (např. tvrdá tužka nefunguje). [8]

3.3.1.4 Technologie využívající infračervené světlo

Princip této technologie spočívá ve vytvoření mřížky infračervených paprsků, tak že se proti sobě umístí dioda vysílající infračervený paprsek a na druhý konec fototranzistor. Stisknutí (přerušení vysílané mřížky) je detekováno uzavřením příslušného fototranzistoru. Pro úsporu energie i opotřebení diod je možné mít aktivovány vysílače pouze v jedné ose a druhou zapnout až v případě detekce dotyku. [9]

Tyto panely jsou těsně uzavřeny a lze je ovládat širokou škálou měkkých i tvrdých předmětů. Díky tomu nachází uplatnění v medicínských a strojních zařízeních. [8]

3.3.2 Výpočet souřadnic místa dotyku

Protože na displeji DT028TFT-TS, který se nachází na kitu, je použita rezistivní technologie s 4drátovým rozhraním, bude tato kapitola věnována postupu pouze pro něj.

Nejdříve je třeba pomocí A/D převodníku zjistit napětí na odporovém děliči (kapitola 3.3.1.1) pro obě osy. Postup je následující:

- Na pin X+ přivedeme napájecí napětí a X- uzemníme. Piny X+ a X- uvedeme do stavu vysoké impedance.
- Na jednom z pinů X+ a X- změříme hodnotu napětí pomocí A/D převodníku.
- Na pin Y+ přivedeme napájecí napětí a Y- uzemníme. Piny Y+ a Y- uvedeme do stavu vysoké impedance.
- Na jednom z pinů Y+ a Y- změříme hodnotu napětí pomocí A/D převodníku.

Protože je poměr vzdáleností k elektrodám přímo úměrný poměru napětí na napěťovém děliči, vzniká nám jednoduchá trojčlenka, ze které vyplývá rovnice 3.1.

$$x = \frac{\text{šířka}_{LCD} * X_{AD}}{AD_{max}}$$
$$y = \frac{\text{výška}_{LCD} * Y_{AD}}{AD_{max}}$$

Rovnice 3.1: Jednoduchý výpočet bodu dotyku

Protože na okrajích displeje není téměř nikdy přesně napájecí napětí nebo zem, je nutné to v tomto výpočtu zohlednit. Displej je nutné nejprve zkalibrovat – tedy zjistit hodnoty napětí pro krajní polohy. Zohlednění těchto hodnot ve výpočtu vidíme v rovnici 3.2.

$$x = \frac{\text{šířka}_{LCD} * X_{AD}}{X_{max} - X_{min}}$$
$$y = \frac{\text{výška}_{LCD} * Y_{AD}}{Y_{max} - Y_{min}}$$

Rovnice 3.2: Výpočet bodu dotyku s kalibračními daty

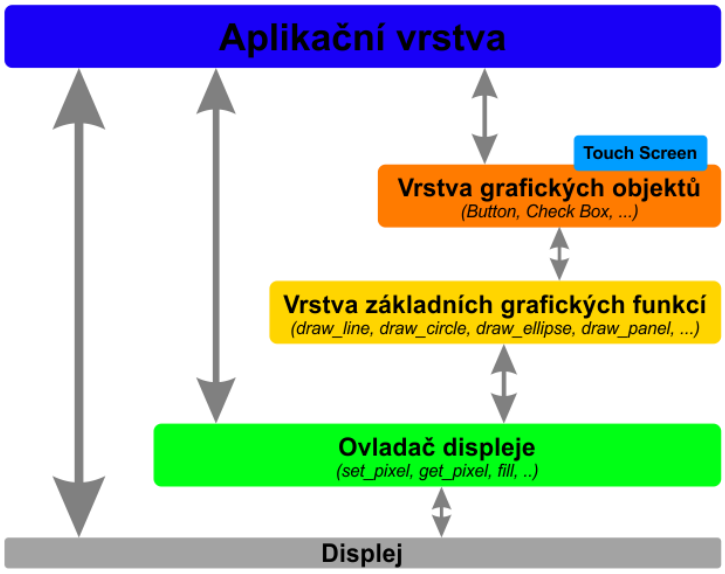
Pokročilejší techniky kalibrace lze nalézt např. v [6]. Pro běžné použití je ale tato základní zcela dostačující.

Poněvadž vodiče tvořící odporovou vrstvu nejsou ideální, mají parazitní kapacitu. Díky ní je třeba počkat, než se výsledné napětí ustálí, a až poté začít měřit hodnotu A/D převodníkem! Uvádí se, že v praxi je vhodné např. provést jedno měření A/D převodníku a jeho výsledek zahodit. Čas potřebný k této operaci by měl být postačující pro ustálení napětí [6]. Pokročilé techniky, které detekují např. stisk displeje jsou zmíněny v implementační části v kapitole 4.3.2.1.

4 Implementace

Grafická knihovna pro dotykový LCD je implementována v jazyce C. Před započetím tvorby knihovny jsem uvažoval o tom, zda by nebylo lepší psát knihovnu v jazyce C++. Ten je narozdíl od jazyka C objektový, což by zpříjemnilo práci s objekty knihovny a umožnilo využití dalších metod jako je např. přetěžování funkcí či zapouzdření. Ačkoli by takto vytvořená knihovna byla jednodušší a příjemnější k použití, její výsledná rychlost by se díky použití těchto pokročilých technik snížila. Vzhledem k tomu, že je kit osazen pouze 8bitovým MCU s 64 kiB paměti, jsem se po konzultaci s vedoucím práce rozhodl použít jazyk C. Navíc IDE (Integrated Development Enviroment) CodeWarrior od firmy Freescale, ve kterém jsem knihovnu vyvíjel, má omezenou výslednou velikost kódu pro C++ překladač na 1 kiB, a protože výsledná velikost knihovny by několikanásobně přesáhla tento limit, bylo by nutné zakoupit nějakou z placených verzí tohoto IDE.

Na obrázku 4.1 vidíme jednotlivé části knihovny a jejich napojení.



Obrázek 4.1: Rozdělení grafické knihovny

4.1 Ovladač displeje

Ke komunikaci s displejem je využito komunikační rozhraní i80/8 bitů, jehož popis lze nalézt v kapitole 3.2.1. Toto rozhraní jsem zvolil proto, že z dostupných nabízí největší přenosovou rychlost. (Samozřejmě jeho varianty na 9, 16 a 18 bitech jsou rychlejší, ale na přídavném modulu je pro tento účel dostupných pouze 8 pinů, protože zbytek je uzemněn. Navíc používáme 8bitový MCU.)

V této vrstvě se nacházejí následující základní funkce pro přenos dat do registrů driveru displeje:

- `void lcd_control_set_reg(byte index)`
  - vybere registr v driveru, do kterého se bude zapisovat

    - je využívána především funkcemi `get_pixel` a `set_pixel` před přenosem dat
- `void lcd_control_write(byte index, word value)`
    - zapíše data (`value`) do registru určeného parametrem `index`

Velice důležitou funkcí, která musí být zavolána vždy po resetu a před prvním použitím knihovních funkcí, je funkce `void lcd_init()`, která provede inicializaci displeje. To zahrnuje např. zapnutí oscilátoru nebo nabití kondenzátorů. Při těchto operacích však bylo nutné mezi ně zařadit zpoždění (např. při čekání na nabití kondenzátorů), se kterým je třeba ve výsledné aplikaci počítat. Celkový čas těchto zpoždění při inicializaci displeje je přibližně 350 ms.

Další důležité funkce této vrstvy:

- `void set_pixel(word x, word y)`
    - nastaví pixel na souřadnicích `x` a `y` na barvu určenou globální proměnnou `__color`
- `word get_pixel(word x, word y)`
    - navrátí hodnotu barvy pixelu na souřadnicích `x` a `y`
- `void fill(word x1, word y1, word x2, word y2)`
    - vyplní obdélník určený souřednicemi `x1, y1, x2, y2` barvou `__color`
    - několika násobně rychlejší než použití funkce `set_pixel` v cyklu
    - určeno i pro kreslení vodorovných nebo svislých čar – zde je zrychlení enormní
- `void __set_color(word color)`
    - nastaví globální proměnou `__color` na hodnotu `color`
- `void __set_color_rgb(byte red, byte green, byte blue)`
    - nastaví globální proměnnou `__color` na barvu spočítanou z proměnných `red`, `green` a `blue`
- `word __get_color()`
    - vrací hodnotu globální proměnné `__color`
- `word make_color(byte red, byte green, byte blue)`
    - navrací hodnotu barvy spočítanou z proměnných `red`, `green` a `blue`

Nastavování barvy přes globální proměnnou jsem zvolil z důvodu minimalizace předávaných parametrů, jejímž důsledkem je úspora času. Např. při kreslení kružnice, kdy se body vykreslují stejnou barvou, by předávání parametrem zbytečně plýtvalo procesorovým časem.

Tuto vrstvu tvoří knihovny `lcd_control.h` a `lcd_pixel.h`.

4.2 Vrstva základních grafických funkcí

Tato vrstva slouží pro kreslení základních geometrických tvarů jako je přímka či elipsa. Obsahuje i funkci pro vykreslení panelu (3D pozadí většiny prvků objektové vrstvy). Součástí této vrstvy jsou i funkce pro vykreslování textu.

Důležité funkce, které jsou obsaženy v této vrstvě:

- `void set_color(word color)`
  - nastaví globální barvu popředí (barvu obrysu) na hodnotu `color`
- `void set_background_color(word color)`
  - nastaví globální barvu pozadí (barvu výplně) na hodnotu `color`
- `void set_line_size(byte size)`
  - nastaví stupeň šířky čáry na `size`
- `void draw_line(int x1, int y1, int x2, int y2)`
  - vykreslí čáru spojující bod [`x1`, `y1`] s bodem [`x2`, `y2`]
  - používá Bresenhamův algoritmus
- `void draw_rectancle(word x1, word y1, word x2, word y2)`
  - vykreslí obdélník daný body [`x1`, `y1`] a [`x2`, `y2`]
- `void draw_circle(int sx, int sy, int r)`
  - vykreslí kruh se středem v bodě [`x`, `y`] a poloměrem `r`
  - základem je Midpoint algoritmus
- `void draw_ellipse(int x, int y, int a, int b)`
  - vykreslí elipsu se středem v bodě [`x`, `y`] a poloosami `a`, `b`
  - základem je Midpoint algoritmus
- `void draw_text(byte* text, word left, word top, word right, word bottom, word color, Font* font, Text_align h_align, Text_align v_align)`
  - funkce vykreslí text do vymezené oblasti (`left`, `top`, `right`, `bottom`) barvou `color`, fontem `font` a zarovná jej dle parametrů `h_align` (horizontálně) a `v_align` (vertikálně)

V této vrstvě se nacházejí i další funkce, o kterých jsem se zde z kapacitních důvodů nezmínil. Často se jedná o variantu některé výše uvedené funkce, která například vykresluje navíc pozadí. Účel těchto funkcí je však jasný již z jejich názvu, případně je popsán v komentáři.

Vrstva je složena z knihoven `low_graphic.h` a `text.h`. Definice písem je obsažena v knihovně `fonts.h`.

4.3 Vrstva grafických objektů

Úkolem této vrstvy je správa grafických objektů. To zahrnuje činnosti od vytvoření objektu až po rozesílání zpráv o událostech na vnějších rozhraních (v tomto případě pouze na dotykovém displeji). Knihovna podporuje vrstvení objektů pomocí vlastnosti z-index (umístění na ose z).

Protože je knihovna napsána v jazyce C, který není objektový, bylo nutné v něm objekty pouze simulovat. Každý objekt je tak ve skutečnosti strukturou. Ta se skládá z následující hlavičky objektu:

```
typedef struct
{
    word left;   // most left pixel (x1)
    word top;    // most top pixel (y1)
```

```
    word right;  // most right pixel (x2)
    word bottom; // most bottom pixel (y2)
    word radius; // radius of corners
    Object_type type;
    int z_index; // when changed call objects_sort()
    Object_state state;
    void (*event_handler)(Object*, Message);
    Color_scheme* scheme;
} Object;
```

Tuto hlavičku obsahuje každý objekt jako první položku ve své struktuře následovanou dalšími specifickými vlastnostmi. Takto pak např. vypadá struktura tlačítka:

```
typedef struct
{
    Object object;
    word text_width;
    word text_height;
    void* text;
} Button;
```

Pokud pak chceme přistupovat k objektu typu tlačítko jako k obecnému objektu, stačí nám použít přetypování. Stejným způsobem to lze provést i opačným směrem. Toto přetypování z obecného objektu na objekt specifický bychom měli používat jen, pokud jistě víme, že objekt je daného typu. Pokud toto nelze zaručit nebo je nám typ předem neznámý, je třeba jeho typ ověřit podle vlastnosti `type` objektu, jinak nám hrozí, že budeme přistupovat k náhodným datům.

Funkce, které tato vrstva obsahuje, je zde zbytečné zmiňovat, protože jsou používány pouze vestavěnými objekty nebo tvoří funkční logiku správce objektů a běžný uživatel k nim nepotřebuje přistupovat. Výjimku tvoří inicializace dotykového rozhraní, která je popsána v kapitole 4.3.2, a metody `void set_event_handler(Object* object, void (*event_handler)(Object*, Message))` a `void erase_event_handler(Object* object)`, které slouží k manipulaci s handlery událostí.

Objektovou vrstvu tvoří knihovna `graphic_object_layer.h`, která zajišťuje správu objektů, `touch.h`, která se stará o dotykové rozhraní, a knihovny objektů (např. `button.h`).

4.3.1 Správce objektů

Správce si trvale uchovává seznam objektů, který je vždy vzestupně seřazen podle vlastnosti z-index. Při všech vnitřních změnách (např. přidání objektu) si správce seznam vždy seřadí sám, pouze kdybychom změnili vlastnost z-index prostým přiřazením, je nutné zavolat metodu `void z_index_changed()`, která by zajistila setřídění seznamu objektů. Toto se silně nedoporučuje používat! Ke změně z-indexu je určena funkce `void change_z_index(Object* object, int z_index)`, která se o vše postará. K řazení objektů jsem použil vylepšený algoritmus Bubble Sort, který je v tomto případě mnohem vhodnější než Quick Sort, který by zbytečně spotřebovával paměť.

Vykreslování objektů probíhá od začátku vzestupně seřazeného seznamu objektů, čímž je dosaženo, že objekty s nejvyšším z-indexem jsou vykreslovány jako poslední a tak překrývají objekty s nižším z-indexem ležící pod nimi. Samozřejmostí je, že správce vykresluje jen ty objekty, které jsou v danou chvíli neplatné. Pokud jsou objekt nebo část obrazovky označeny za neplatné, správce označí k překreslení všechny objekty, kterých se tato změna přímo týká.

Další úlohou správce je distribuce přijatých událostí objektům, na kterých vznikly. Takto se každá událost dostane pouze jedinému objektu, který je na souřadnicích vzniku v hierarchii objektů nejvýše (má nejvyšší z-index). Objekt dle předdefinovaného chování na událost zareaguje (např. změní svojí barvu pozadí), a pokud uživatel nastavil objektu handler (obslužnou metodu) události, objekt zavolá i tuto uživatelovu metodu.

4.3.2 Dotykové rozhraní

Dotykové rozhraní slouží k detekci a zpracování událostí vzniklých na dotykové vrstvě displeje. Pokud chceme toto rozhraní používat, je nutné zavolat funkci `void touch_init()`, která aktivuje ovladač dotykové vrstvy.

4.3.2.1 Ovladač

Princip zjištění souřadnic na 4drátovém odporovém displeji byl již popsán v kapitole 3.3.1.1. V této kapitole se tedy budu věnovat pouze rozšiřujícím technikám či algoritmům, které jsem v ovladači implementoval.

První z těchto technik, které bylo potřeba implementovat, je detekce stisku na dotykovém displeji. Pro toto existují v zásadě dvě techniky. První z nich provádí periodickou kontrolu, zda není stisknuta dotyková vrstva, a druhá využívá přerušení pomocí modulu KBI. V knihovně jsem implementoval variantu, která využívá KBI modul, protože má rychlejší odezvu a nezatěžuje jádro mikrokontroléru periodickým prováděním detekce stisku. Princip detekce stisku pomocí modulu KBI je tedy následující [6]:

- Nastavíme piny X-, Y+ do stavu vysoké impedance
- Pin X+ uzemníme
- Povolíme „pull-up“ rezistor na pinu Y-, který nám bude definovat napěťovou úroveň logické „1“, pokud nebude Y- nikam připojen (nebudou spojeny vrstvy X a Y)
- Nastavíme KBI modul na pin Y- s přerušením při sestupné hraně

Zvýše uvedeného postupu vyplývá, že pokud uživatel stiskne dotykovou vrstvu, vrstva X a Y se spojí a pin Y- se tak uzemní. Tím vzniká sestupná hrana, kterou KBI modul zachytí a vyvolá přerušení. Knihovna se snaží zabránit vyvolání falešných událostí stisku dotykové vrstvy, a tak v obsluze tohoto přerušení ještě jednou kontroluje, zda je dotyková vrstva stále ve stisknutém stavu [6].

Poté, co jsme úspěšně detekovali stisk displeje, nastavíme časovač TPM, aby nám po 20 ms generoval přerušení, v jehož obsluze provedeme zjištění souřadnic pomocí A/D převodníku, jak je popsáno v 3.3.2. Zde je důležité výsledky prvního měření zahodit, čímž mezitím docílíme ustálení napětí. Knihovna poté provádí pět měření (v 12bitovém režimu ADC), ze kterých vybere výslednou hodnotu jako medián [6]. Tím je docíleno větší přesnosti a zbavení se náhlých odchylek v měření.

Posledním úkolem ovladače je detekce zániku stisku (událost „pen-up“) dotykové vrstvy. Tato funkce je zajištěna kontrolou před každou konverzí a probíhá následovně:

- Nastavíme piny X+, Y+ do stavu vysoké impedance
- Pin X- uzemníme
- Povolíme „pull-up“ rezistor na pinu Y-, který nám bude definovat napěťovou úroveň logické „1“, pokud nebude Y- nikam připojen (nebudou spojeny vrstvy X a Y)
- Počkáme minimálně 0,5 ms, než se napětí ustálí (hodnota byla zjištěna experimentálně)

Pokud na pinu Y- bude hodnota logické „1“, znamená to, že se vrstvy nedotýkají a je třeba ukončit periodické vzorkování (vypnout TPM a ADC) a vyvolat událost `ev_touch_up`.

Ovladač dotykové vrstvy generuje následující tři události:

- `ev_touch_down`
  - je vyvolána při stisku dotykové vrstvy
- `ev_touch_move`
  - je vyvolána při pohybu po dotykové vrstvě
- `ev_touch_up`
  - je vyvolána, pokud není displej stisknut a v předchozím cyklu byl

Ty pak ovladač zasílá správci objektů, který podle připojených souřadnic vzniku zjistí, kterému objektu zpráva o události patří a přepošle mu jí. Zde je nutné pamatovat na to, že po dobu obsluhy přerušení, tedy i uživatelských funkcí (handlerů), které jsou na tyto události napojeny, nelze detekovat další události. Je tedy dobré v nich nevykonávat náročné výpočty, které by zhoršovaly odezvu tohoto rozhraní.

Celý ovladač dotykové vrstvy je založen pouze na soustavě přerušení, takže spotřebovává jen minimum procesorového času.

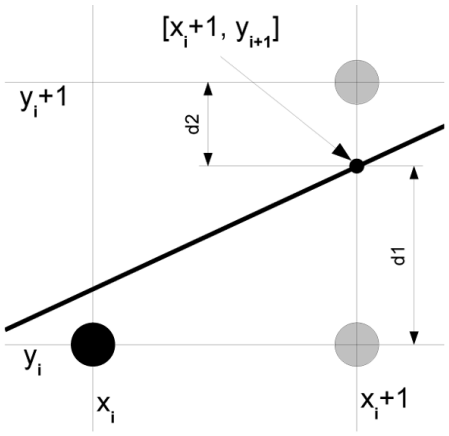
4.4 Použité grafické algoritmy

V následujících podkapitolách jsou zmíněny některé důležité grafické algoritmy, které jsem v knihovně implementoval. Tato kapitola slouží především jako náhled do problematiky a vysvětlení základních principů těchto algoritmů. Na jejich implementaci lze nahlédnout do zdrojových kódů knihovny. Zdrojem informací o Bresenhamovu algoritmu pro kreslení čáry a Midpoint algoritmu je [10]. Zde můžete také nalézt podrobnější informace o odvození rovnic pro výpočty.

4.4.1 Bresenhamův algoritmus kreslení čáry

Algoritmus publikoval J. E. Besenham už v roce 1965. Tento algoritmus jsem pro kreslení čáry zvolil proto, že je v dnešní době nejpoužívanější a je nejefektivnější. Pracuje s celými čísly a využívá pouze operace sčítání, odčítání a porovnávání. Díky tomu je jeho implementace v hardwaru relativně snadná.

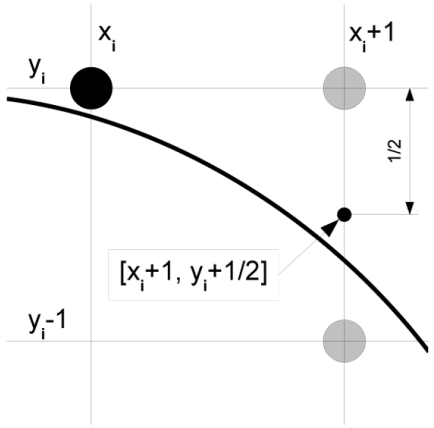
Princip tohoto algoritmu vychází z obecné rovnice přímky a zjištění, blíže kterému bodu prochází přímka. Tato myšlenka je znázorněna na obrázku 4.2.



Obrázek 4.2: Princip Bresenhamova algoritmu

4.4.1 Midpoint algoritmus

Ačkoli by šlo pro kružnici odvodit Bresenhamův algoritmus podobně jako pro úsečku, zvolil jsem pro vykreslování kružnice a elipsy Midpoint algoritmus, protože je častěji používaný. Jeho princip je v podstatě stejný a liší se pouze podmínkou pro stanovení prediktoru. Princip rozhodovaní se, který bod je kružnici blíže, znázorňuje obrázek 4.3.



Obrázek 4.3: Princip Midpoint algoritmu pro kružnici

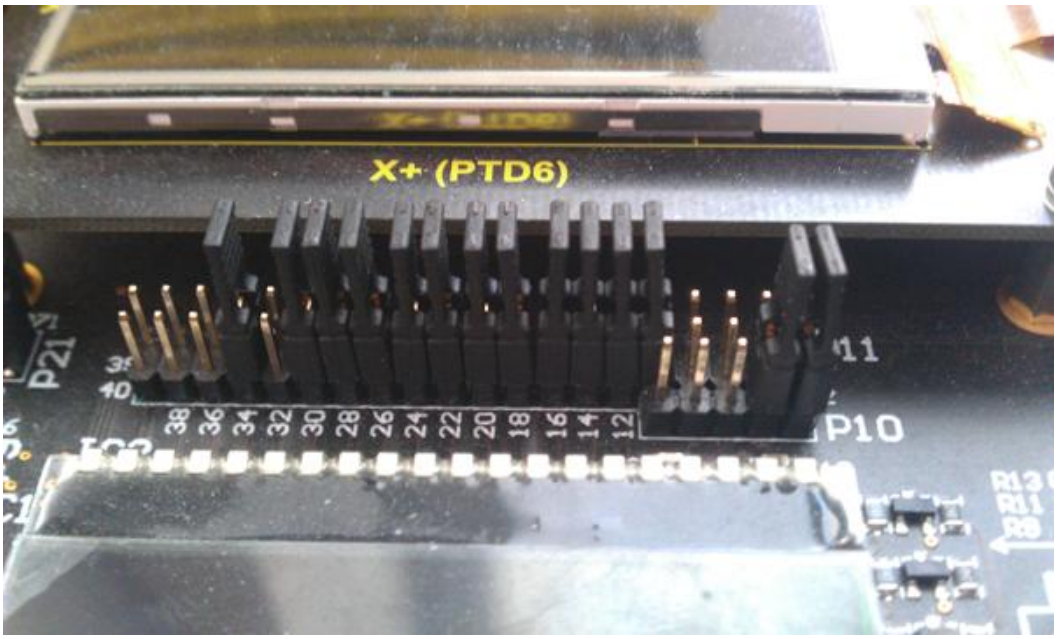
4.4.2 Algoritmus pro zneplatňování objektů

Vstupem jednoho kroku algoritmu je obdélníková oblast označená k zneplatnění. Algoritmus prochází vzestupně seřazené objekty od konce seznamu a zkoumá, zda a jak se takto vyznačená oblast s daným objektem protíná. Určení vzájemné polohy je založeno na principu algoritmu Cohen-Sutherlend [10] , který dle předem spočítaných kódů určí vzájemnou polohu a rozdělí si objekt na podoblasti. Poté už jen rekurzivně zavolá sám sebe na oblasti, které nebyly tímto objektem v zadaném čtverci pokryty, a pokud nalezl průnik objektu s danou oblastí, označí jej k překreslení. Takovýto přístup k řešení problému je nazýván algoritmem „Rozděl a panuj“.

5 Návod k použití

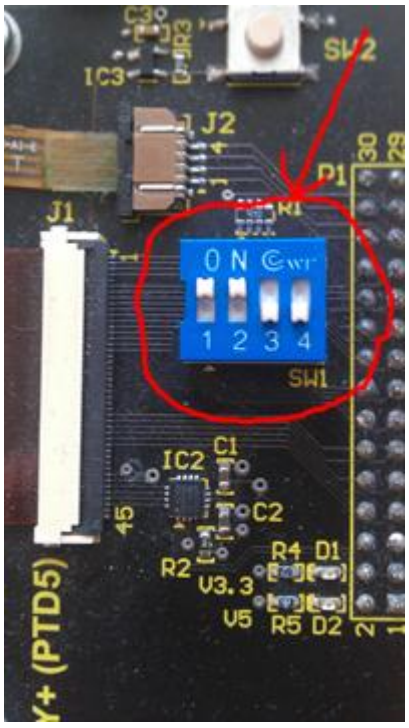
Hned na úvod by bylo dobré zmínit, které moduly a porty knihovna využívá, a předejít tak budoucím kolizím s uživatelským kódem. Uživatel nesmí použít port B, na kterém probíhá datový přenos mezi MCU a driverem displeje, piny 0-5 portu A, které slouží pro řízení přenosu a ovládání podsvícení, pin 5 portu F, který řídí podsvícení, a pokud používá dotykovou část knihovny, tak ani piny 3-6 na portu D. Při zapnuté podpoře dotykové vrstvy nelze využít časovač TMP2 a přerušení na modulu KBI a jeho pin č. 3.

Když už víme, které součásti je třeba nepoužívat, můžeme přejít k nastavením na kitu, bez kterých nemůže knihovna s displejem komunikovat. Nejprve je třeba na P11 spojit zkratovacími propojkami následující dvojice pinů: 9-10 až po 29-30 a 33-34. Dále je nutné spojit porty P11 a P10 takto: pin 2 (P11) s pinem 1 (P10) a pin 4 (P11) s pinem 2 (P10). Zbylé piny nejsou propojeny. Tímto jsme zajistili, že se řídící signály dostanou z portů MCU k modulu displeje. Toto zapojení pro přehlednost zobrazuje obrázek 5.1.



Obrázek 5.1: Propojení pinů na portu P11 a P10

Posledním důležitým nastavením je nastavení komunikačního rozhraní. Jak bylo zmíněno v kapitole 4.1, ke komunikaci s driverem displeje používá knihovna rozhraní i80/8 bitů, které je třeba nastavit kombinací „1100“ na IM[0-3]. To lze provést nastavením přepínače SW1 přídavného modulu na stejnou hodnotu. Nastavení je znázorněno na obrázku 5.2.



Obrázek 5.2: Nastavení komunikačního rozhraní i80/8 bitů

5.1 Začátek práce s knihovnou

Nejdříve je třeba překopírovat soubory knihovny do projektu, případně k nim nastavit cestu a zahrnout je do překladu. Nejjednodušší variantou je převzít projekt knihovny, kde je již vše patřičně nalinkované a upravit ho (např. přejmenovat atd.). Jako první je třeba přidat následující kód, který přidává všechny potřebné knihovní soubory:

```
#include "graphic.h"
```

Před použitím knihovních funkcí je třeba knihovnu inicializovat voláním následující funkce:

```
lcd_init();
```

Pokud chceme na displeji využívat dotykovou vrstvu, je nutné přidat ještě následující řádek:

```
touch_init();
```

To je vše, co musíme před použitím knihovny udělat.

5.2 Práce s vybranými funkcemi ovladače

1. **Nastavení barvy a vykreslení pixelu:**

```
word cervena = make_color(255, 0, 0);
__set_color(cervena);
set_pixel(10, 10);    // červený bod
set_pixel(11, 11);    // červený bod
__set_color_rgb(0, 255, 0);      // zelená
set_pixel(10, 11);    // zelený bod
set_pixel(11, 10);    // zelený bod
```

2. **Výplň a její použití k akceleraci vodorovných a svislých čar:**

```
word cervena = make_color(255, 0, 0);
set_background_color(cervena);
fill(0, 0, 319, 239);
__RESET_WATCHDOG();  // pozor - přenáší se velké množství dat
set_background_color_rgb(0, 255, 0);  // zelená
fill(0, 119, 319, 119);    // vodorovná dělicí čára
set_background_color_rgb(0, 0, 255);  // modrá
fill(159, 0, 159, 239);    // svislá dělicí čára
```

Zde je vhodné upozornit na to, že velké přenosy do driveru displeje trvají nezanedbatelný čas a může se stát, že nám po čase začne hlídací obvod resetovat MCU!

5.3 Práce se základními grafickými funkcemi

Protože již víme, jakým způsobem jsou funkce knihovny pojmenovány a jak se s nimi pracuje, následující ukázka práce s funkcemi této vrstvy snad bude vypovídající sama o sobě. Samozřejmě jsou zde uvedeny jen některé význačné funkce, protože jejich zbytek je pojmenován stejným stylem a postup práce je analogický. Jedinou výjimku zde tvoří funkce `draw_char`, jejíž barva vykreslování se nastavuje pomocí funkce `__set_color`. Tento odlišný přístup jsem zvolil z hlediska výkonnosti. Běžný uživatel jí ani pravděpodobně používat nebude a využije pokročilejší varianty, která je implementována funkcí `draw_text`.

```
//nastavíme tloušťku kreslené čáry (obrysu)
set_line_size(1);

cervena = make_color(255, 0, 0);
set_color(cervena);
draw_line(0, 0, 319, 239);

set_color_rgb(0, 255, 0);
draw_line(0, 239, 319, 0);

set_color_rgb(255, 255, 255);
draw_circle(144, 119, 20);
set_background_color_rgb(0, 0, 255);
draw_circle_filled(174, 119, 20);

__set_color_rgb(255, 255, 0);
draw_char('a', 159, 113, &Comic_sans_ms);

set_color_rgb(150, 150, 150);
draw_rectancle(100, 50, 150, 100);
```

```
draw_text("muj napis", 100, 50, 150, 100, make_color(0, 255,
255), &Comic_sans_ms, ta_h_center, ta_v_top);
```

5.4 Práce s objekty

Protože práce s objekty jako takovými je pro všechny stejná, v následující ukázce bude použit typ `Button`, tedy tlačítko. Ostatní objekty mají některé vlastnosti specifické a zde mi nezbývá než odkázat na přiloženou dokumentaci, ale i bez ní se dá účel těchto vlastností pochopit z jejich názvů. Všechny ale obsahují na svém začátku strukturu `Object`, jak bylo vysvětleno v kapitole 4.3.

Protože každý objekt potřebuje používat několik barev (obrys, pozadí, text ad.), bylo nutné zavést barevné schéma. To je struktura, ve které jsou uloženy barvy částí objektu pro různé stavy. Knihovna obsahuje nadefinované výchozí schéma, není ale problém jej modifikovat, upravit nebo na něm založit schéma nové.

```
// založíme naše schémata na výchozím
#include <string.h>    // obsahuje memcpy2
memcpy2(&cs[0], &default_color_scheme, sizeof(Color_scheme));
memcpy2(&cs[1], &default_color_scheme, sizeof(Color_scheme));

// nastavíme jejich specifické vlastnosti
cs[0].text_color = make_color(255, 0, 0);
cs[0].background_color = make_color(130, 130, 130);

cs[1].emboss_light_color = make_color(255, 255, 255);
cs[1].background_color = make_color(0, 200, 0);

// vytvoříme objekty
button1 = button_create(20, 20, 120, 52, 50, &cs[0], 0, "muj
popisek");
// použijeme radius 10 a zaoblíme tak rohy panelu
button2 = button_create(30, 100, 150, 120, 10, &cs[1], 10,
NULL);
// pokud uvedeme NULL místo adresy schématu, bude použito
výchozí schéma
button3 = button_create(20, 10, 70, 130, 2, NULL, 0, NULL);

// pokud je již nepotřebujeme, můžeme je zrušit
button_destroy(button1);
button_destroy(button2);
button_destroy(button3);
```

Knihovna nabízí možnost rozšíření počtu objektů o naše vlastní. Jedná se o „prázdný“ objekt, který nám dovolí nastavit vlastní metodu vykreslování, zpracování událostí i uchovávání specifických vlastností. Takovýto objekt sice nebude objektem ve smyslu základních objektů, ale lze jím nasimulovat stejná činnost. Tyto funkce, jsou nabízeny objektem `User_object`.

```
void kresli(Object* object)
{
	set_color_rgb(0, 100, 255);
	draw_circle(100, 100, 50);
}

User_object* user_object = user_object_create(0, 0, 100, 100,
10, NULL);
user_object_set_draw_handler(user_object, &kresli);
```

Pokud změníme nějakou vlastnost objektu, která ovlivňuje jeho vzhled nebo obsah, je třeba objekt zneplatnit a vynutit tak jeho překreslení. Knihovna automaticky překreslí i objekty, do kterých tato změna zasáhne. Postup následuje níže.

```
button2->text = "novy popisek";
invalidate_object((Object*)button2);
repaint();
```

Kdybychom chtěli objekt na obrazovce přemístit, je třeba myslet na to, že je třeba zneplatnit místo o velikosti objektu, odkud objekt přesouváme, a také místo, kam objekt přesuneme. Tuto práci si můžeme ušetřit použitím knihovní funkce void object_move_to(Object* object, word x, word y).

```
object_move_to((Object*)button2, 60, 80);
repaint();
```

Pokud takto nějaký objekt přemístíme nebo jej zrušíme, zjistíme, že část objektu, která ležela pouze na fiktivním pozadí (vykresleného např. pomocí funkce `fill`), zůstala na displeji. Tato část již není nijak aktivní a nereaguje na vstupní události. To je způsobeno tím, že na daném místě není žádný objekt, kterým by se tato část překreslila. Při implementaci mě napadli 2 řešení. V prvním z nich by si knihovna vytvořila pomocný objekt pozadí a nastavila mu nejnižší z-index. Druhé řešení, které je v knihovně implementováno, nabízí uživateli objekt Background, který si uživatel vytvoří a smí s ním manipulovat. To může být v některých situacích k užitku a navíc je filozofií knihovny, že je ovládána uživatelem a, pokud je to technicky možné, měl by to být on, kdo má možnost nastavení co možná nejvíce věcí a ne pouze jejich podmnožinu. Následující postup ukazuje způsob práce s pozadím.

```
background = background_create(make_color(0, 0, 0));
__RESET_WATCHDOG();
```

Pokud by bylo při použití knihovny vytvořeno velké množství objektů, které by se vzájemně v několika vrstvách překrývali, bylo by vhodné přizpůsobit (zvětšit) čas hlídacího obvodu, aby měla knihovna šanci, stihnout takto složitou scénu vykreslit.

5.5 Seznam objektů

- Background (pozadí)

- Button (tlačítko)
- Check box (zaškrtávací pole)
- Combo box (výběrové pole)
- Label (popisek)
- Progress bar (ukazatel postupu)
- Radio button (přepínací pole)
- Slider (posuvník)
- Text box (vstupní jednořádkové pole)
- Text area (vstupní víceřádkové pole)
- User object (uživatelský objekt)

6 Závěr

Cílem mé práce bylo vytvořit na výukovém kitu grafickou knihovnu pro dotykový LCD. Knihovnu jsem rozšířil např. o možnost použití více vrstev (z-index) a možnost vytvořit uživatelský objekt.

V porovnání s jinými knihovnami, konkrétně s Microchip Graphic Library, je moje knihovna bohatší o podporu vrstvení objektů. Na druhou stranu tato knihovna je bohatší ve své výbavě (množství objektů) a podporovaných řadičích displeje. Nabízí i pokročilejší grafiku a práci s obrázky.

Jako možné rozšíření do budoucna vidím právě přidání podpory práce s obrázky, rozšíření spektra fontů i jejich velikostí. Z grafického hlediska by byly užitečné funkce pro vykreslování např. přechodem, namísto vykreslování pouze barvou. Pokud by byla knihovna přenesena na výkonnější mikrokontrolér, bylo by dobré uvažovat o podpoře základních animací či průhlednosti.

Můj přínos shledávám v možnosti využít tuto knihovnu ve výuce hardwarových předmětů jako je IMP (Mikroprocesorové a vestavěné systémy), kde na tomto kitu probíhá výuka ve cvičeních.

Jelikož jsem nikdy nic podobného nedělal, přínos této práce je pro mě velký. Získal jsem spoustu cenných znalostí z oblasti programování MCU a vestavěných systému jako takových.

Literatura

[1] *MC9S08JM60: Data Sheet*. Freescale Semiconductor Inc., 2009. 388 s.

[2] Pereira, Fabio: *HCS08 Unleashed: Designer's Guide To The HCS08 Microcontrollers*. 2. vyd. USA: Charleston, SC, 2010, 300 s. ISBN 1-4196-8592-9

[3] *LCD MODULE DT028TFT-TS: Product Specification*. Displaytech Ltd., 2008. 11 s.

[4] *ILI9320: Datasheet*. ILITEK, 2008. 109 s.

[5] Šimek, Václav: *Schéma výukového kitu*. FIT VUT, Brno

[6] Juengst, Andy: *Resistive Touchscreen Controller Using the S08 Family*. Freescale Semiconductor Inc., 2010. 11 s.

[7] Půlpán, Radek: *Analogové touchscreeny a jejich použití* [online], poslední revize 9. 4. 2004 [cit. 4. 5. 2011].
URL http://hw.cz/Teorie-a-praxe/Dokumentace/ART1086-Analogove-touchscreeny-a-jejich-pouziti.html

[8] *Technologie dotykových panelů* [online]. [cit. 4. 5. 2011].
URL http://www.vekobs.cz/dotykove-technologie

[9] Sekanina, Pavel: Vývojová deska s mikrokontrolérem a displejem touchscreen, Brno, FIT VUT v Brně, 2010

[10] Kršek, Přemysl: *Základy počítačové grafiky: Studijní opora*. FIT VUT, Brno. Verze 0.9

Seznam příloh

Příloha 1. CD

Obsah CD

- Zdrojové kódy knihovny
- Ukázková aplikace (projekt prostředí CodeWarrior)
- Dokumentace
- Text práce ve formátu .docx a .pdf