VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

KNIHOVNA FUNKCÍ PRO OBSLUHU PERIFERIÍ NA VÝUKOVÉM KITU S MCU FREESCALE

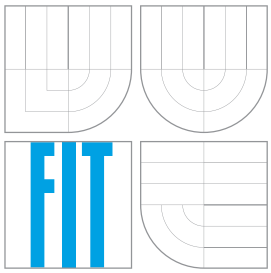
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PAVEL GRUNT

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

KNIHOVNA FUNKCÍ PRO OBSLUHU PERIFERIÍ NA VÝUKOVÉM KITU S MCU FREESCALE

LIBRARY OF SERVICE ROUTINES FOR PERIPHERALS ON EDUCATIONAL KIT

WITH FREESCALE MCU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PAVEL GRUNT
AUTHOR

VEDOUCÍ PRÁCE — Ing. VÁCLAV ŠIMEK
SUPERVISOR

BRNO 2012

Abstrakt

Práce se zabývá návrhem a implementací knihovny funkcí pro obsluhu periferií na výukovém kitu s mikrokontrolery Freescale. Představuje kit, vlastnosti jednotlivých periferních zařízení a způsoby komunikace s nimi. Dále se věnuje vytvoření řadiče dotykového displeje na přídavném modulu kitu a jeho použití v grafické knihovně Freescale eGUI.

Abstract

The thesis deals with designing and implementation of a library of service routines for peripherals on educational kit with Freescale microcontrollers. It presents the laboratory kit, features of individual peripherals and ways of comunication with them. Further, the thesis describes developing the driver for LCD touchscreen on auxiliary kit module and its use in graphic library Freescale eGUI.

Klíčová slova

knihovna, kit, periferie, mikrokontroler, HCS08, ColdFire, ILI9320, řadič displeje, Freescale, eGUI, D4D

Keywords

library, kit, peripheral, microcontroller, HCS08, ColdFire, ILI9320, LCD controller, Freescale, eGUI, D4D

Citace

Pavel Grunt: Knihovna funkcí pro obsluhu periferií na výukovém kitu s MCU Freescale, bakalářská práce, Brno, FIT VUT v Brně, 2012

Knihovna funkcí pro obsluhu periferií na výukovém kitu s MCU Freescale

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Václava Šimka.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Grunt
15. května 2012

Poděkování

Rád bych poděkoval panu Ing. Václavu Šimkovi za rady, pomoc a náměty při vývoji bakalářské práce.

Obsah

Kapitola 1

Úvod

S mikrokontrolery, periferními zařízeními a vestavěnými systémy obecně se v dnešní době setkáváme takřka všude. Aby bylo vytváření zařízení s periferiemi snazší, je vhodné používat knihovny funkcí, které usnadní programátorovi práci a urychlí vývoj.

Cílem práce bylo právě takovou knihovnu vytvořit. Knihovna pokrývá periferie laboratorního kitu, implementuje komunikační protokoly jednotlivých zařízení a zprostředkovává základní obsluhu jejich funkcí.

Nad rámec zadání jsem se věnoval zprovoznění grafické knihovny na rozšiřujícím modulu výukového kitu.

První kapitola seznamuje čtenáře s výukovým kitem. Představuje používané mikrokontrolery a jejich programovací model.

V druhé kapitole se dozvídáme bližší informace o jednotlivých periferiích. Je zde vysvětlen jejich účel, způsob komunikace s mikrokontrolerem.

Třetí kapitola ukazuje strukturu knihovny, použité implementační prostředky. V této kapitole také nalezneme aplikační rozhraní knihovny a možnosti konfigurace.

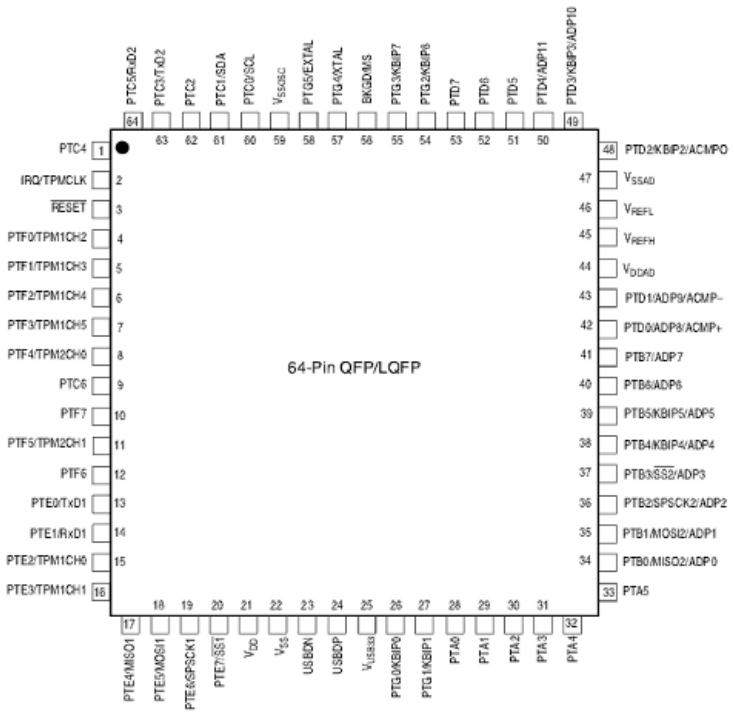
Ve čtvrté kapitolr se dozvídáme informace o grafické knihovně Freescale eGUI. Popisuje vlastnosti a schopnosti této knihovny. Věnuje se rozšiřujícímu modulu a vysvětluje, jak probíhá komunikace s řadičem displeje.

Kapitola 2

Výukový kit

Laboratorní kit slouží pro podporu výuky hardwarových předmětů. Základní jednotkou kitu je mikrokontroler osazený do 64 pinového pouzdra (viz 2.1). K dispozici jsou mikrokontrolery značky Freescale: MC9S08JM60 [1] a MCF51JM128 [2].

Na kitu je 12 MHz krystal, který může sloužit jako zdroj hodinového signálu. Mikrokontroler je možné programovat pomocí externího programátoru (např. P&E Multilink/Cyclone Pro) nebo prostřednictvím USBDM [3] programátoru založeném na mikrokontroleru MC9S08JS16. Výukový kit je možno napájet z externího programátoru nebo přes USB konektor či pomocí adaptéru napětím 5V.



Obrázek 2.1: Mikrokontroler MC9S08JM60 v 64 pinovém pouzdru (převzato z [1])

2.1 Mikrokontroler MC9S08JM60

Mikrokontroler je založen na 8 bitovém jádru HCS08. Maximální frekvence, na níž může pracovat, je 48 MHz. Jeho vnitřní sběrnice pracuje vždy na poloviční frekvenci jádra. Mikrokontroler disponuje 60 KiB flash paměti a 4 KiB RAM paměti. Z vnitřních periferií obsahuje:

- Analogový komparátor
- Analogově-číslicový převodník – dvanácti kanálový převodník s 12 bitovým rozlišením
- MCG – modul pro nastavení zdroje hodin mikrokontroleru
- KBI – osmi kanálový modul přerušení klávesnice
- TMP – dva moduly časovače/pulsně šířkové modulace s šesti, respektive dvěma kanály
- SCI – dva nezávislé moduly asynchronní sériové komunikace
- SPI – dva nezávislé moduly synchronní sériové komunikace mikrokontroleru s periferními zařízeními, rozhraní je plně duplexní – komunikace probíhá oběma směry.
- IIC – modul pro synchronní sériovou komunikaci s dalšími integrovanými obvody
- USB – modul založený na specifikaci USB 2.0 s komunikací v režimu Full-speed (12 Mbps)

Programátorovi je přístupná sada registrů:

- Akumulátor (A) – Jedná se o 8 bitový registr pro obecné užití, řada instrukcí jej využívá pro uložení výsledku.
- Index Registr (H:X) – Ve skutečnosti je to dvojice 8 bitových registrů, které pracují společně jako 16 bitový registr. Toto rozdělení je z důvodů zpětné kompatibility kódu s rodinou mikrokontrolerů M68HC05.
- Ukazatel zásobníku (SP) – Obsahem tohoto 16 bitový registru je adresa vrcholu zásobníku.
- Programový čítač (PC) – Jedná se o 16 bitový registr obsahující adresu instrukce, která bude provedena v dalším kroku.
- Stavový registr (CCR) – Osmi bitový registr nastavuje masku přerušení a obsahuje různé příznaky.

2.2 Mikrokontroler MCF51JM128

Jedná se o 32 bitový mikrokontroler, jehož frekvence dosahuje až 50,33 MHz. Od mikrokontroleru MC9S08JM60 se liší především tím, že je založen na jádru V1 ColdFire a velikostí paměti – má k dispozici 128 KiB flash paměti a 16 KiB paměti RAM. Tento mikrokontroler byl navržen, aby byl kompatibilní s 8 bitovými mikrokontrolery. Periferie na čipu jsou namapovány na stejné porty jako v případě MC9S08JM60.

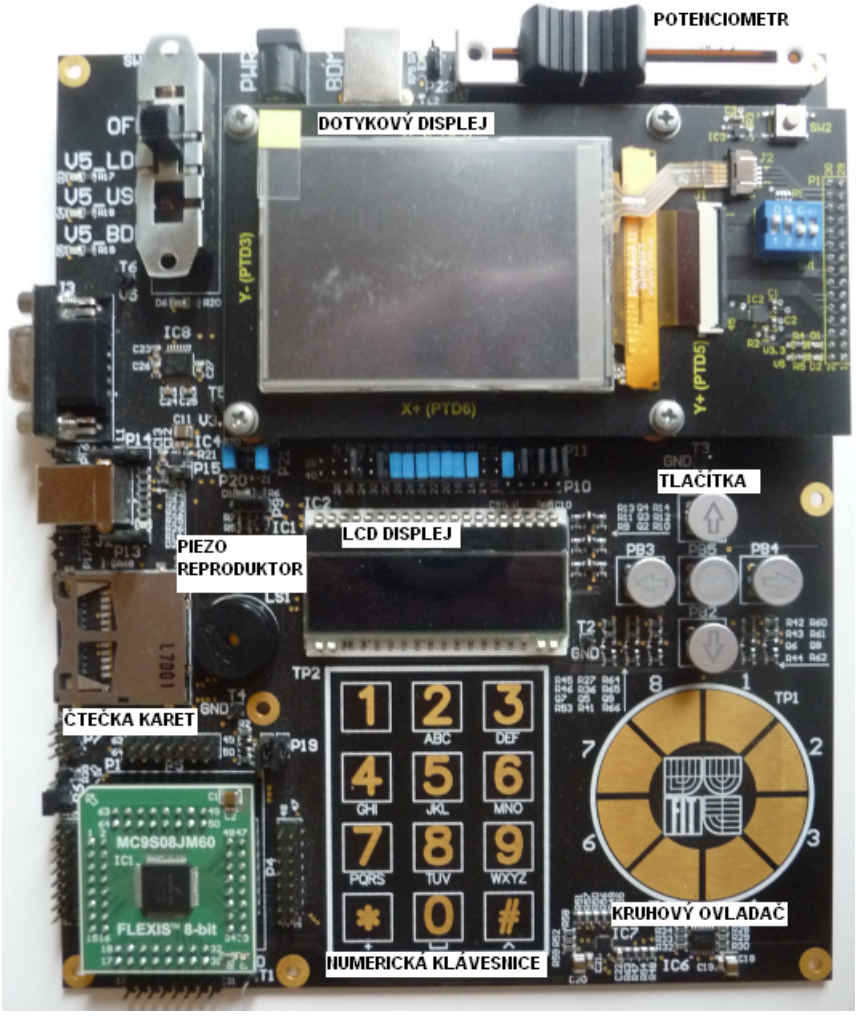
Programovací model architektury ColdFire je rozdělen na uživatelský a privilegovaný. Registrová struktura je oproti HCS08 bohatší:

- Datové registry (D0-D7) – Jádro ColdFire programátorovi nabízí celkem osm 32 bitových registrů. Mohou být použity jako operandy i jako index registry.
- Adresní registry (A0-A6) – Tyto 32 bitové registry slouží především jako index registry, ukazatele apod.
- Ukazatel zásobníku (A7) – Ve skutečnosti architektura ColdFire podporuje dva nezávislé ukazatele zásobníku – pro uživatelský i pro privilegovaný přístup – v závislosti na tom, v kterém módu pracujeme, se nám jeden ukazatel zásobníku jeví jako `A7` a druhý jako `OTHER_A7`.
- Programový čítač (PC) – Registr, jehož hodnotou je adresa prováděné instrukce.
- Stavový registr – V privilegovaném módu se jedná o 16 bitový registr, v uživatelském módu je přístupná pouze spodní polovina, kterou je registr CCR.
- Registr adresy vektoru vyjímek (VBR) – Jak z názvu vyplývá, jedná se o registr, jehož hodnotou je počáteční adresa tabulky vyjímek. Tento 32 bitový registr je přístupný pouze v privilegovaném m odu.
- Registr konfigurace CPU (CPUCR) – S privilegovaným přístupem může programátor konfigurovat různé vlastnosti CPU.

Kapitola 3

Přehled periferií

Informace o jednotlivých periferiích jsem získal z katalogových listů a schématu [4].



Obrázek 3.1: Výukový kit

3.1 Tlačítka

Na kitu je sada pěti tlačítek. Čtyři z nich mají symbol šipky, na pátem je nápis „OK“. Tlačítka jsou připojena k modulu KBI, který generuje přerušení při stisku tlačítka.

Na pinech s tlačítky je pull-up rezistor, který na pinu udržuje hodnotu logické 1. Při stisknutí tlačítka dojde ke spojení se zemí, což vede na detekci sestupné hrany modulem KBI, který vyvolá přerušení.

Směrová tlačítka se mohou používat například pro pohyb v menu, mohou sloužit pro posun kurzoru, tlačítko „OK“ potvrzování volby.

3.2 Potenciometr

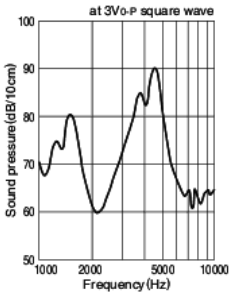
Na kitu je umístěn lineární tahový potenciometr, skládá se z odporové dráhy, po níž se pohybuje jezdec.

Potenciometr je připojen k AD převodníku mikrokontroleru. Funguje jako regulovatelný odporový napěťový dělič.

V současnosti se nevyužívá analogových vlastností potenciometru. Ve spojení s AD převodníkem dochází k převodu analogového výstupu na digitální a s převedenou hodnotou můžeme libovolně pracovat.

3.3 Piezo reproduktor

Jedná se o reproduktor modelu PS1740P02 firmy TDK. Vyznačuje se vysokou hlasitostí generovaných tónů (viz 3.3). Tento typ reproduktoru se používá pro vydávání výstražného či upozorňovacího tónu v domácích spotřebičích (telefony, ledničky, klimatizace).



Obrázek 3.2: závislost hlasitosti tónu na frekvenci (převzato z [5])

3.3.1 Princip generování tónu

Pro generování tónu o zvolené frekvenci se využívá modulu časovače mikrokontroleru. Ten je nutné nastavit, aby pracoval ve srovnávacím výstupním módu, tento princip byl popsán v [6]. Pro vytvoření signálu o zvolené frekvenci je nutné v polovině periody signálu provést inverzi hodnoty pinu, na nějž je reproduktor připojen. Toho se dosáhne nastavením pracovního registru časovače na odpovídající hodnotu (výpočet této hodnoty viz rovnice 3.1).

Jelikož výpočty potřebných hodnot a nastavení příslušných registrů zabírají určitý čas, je reálná frekvence tónu zkreslena (odchylka se pohybuje v řádu desetin procenta). Proto, aby bylo zkreslení nejmenší, je nutné nastavit frekvenci hodin procesoru na maximální hodnotu, čímž se zkrátí doba potřebná pro výpočty a nastavování registrů.

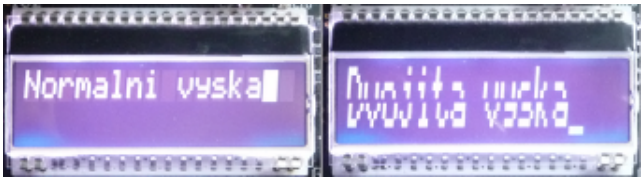
$$Hodnota\ pracovního\ registru = \frac{\frac{f_{časovače}}{f_{tónu}}}{2} \tag{3.1}$$

3.4 LCD displej

LCD displej EA DOG-M 162[7] slouží pro zobrazovaní 16 znaků na každém řádku.
Je ovládán řadičem ST7036[8].

Řadič disponuje pamětí pro zobrazení 80 znaků. Při použití displeje EA DOG-M 162 je viditelných 16 znaků na každém řádku. Pro zobrazení znaků, které leží mimo viditelnou oblast, má řadič instrukce, které posouvají obrazem displeje. Díky tomu je možné zobrazit všechny znaky uložené do paměti. Řadič obsahuje generátor pro 256 znaků (rozměr jednoho znaku je 5 sloupců, 8 řádků), z nich je 248 předdefinováno a zbylých 8 si může uživatel vytvořit. Další zajímavou vlastností je možnost použít jednořádkový mód a dvojnásobnou velikost fontu. Rozdíl je vidět na obrázku 3.3.

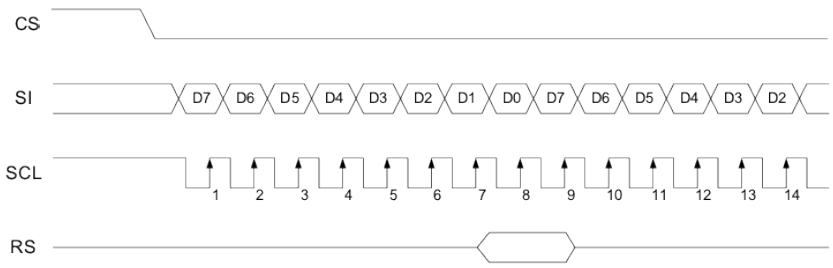
Displej má RGB podsvícení. Odstín jednotlivých barev je možné nastavovat pomocí pulzně šířkové modulace.



Obrázek 3.3: Srovnání velikosti fontu

3.4.1 Komunikace

Řadič displeje umožňuje komunikaci přes 4 nebo 8 bitové paralelní rozhraní nebo přes SPI rozhraní. Na výukovém kitu je zvolena varianta komunikace pomocí SPI rozhraní (viz 3.4). Pro umožnění komunikace je nutné propojit piny na portu P11 (dvojice 1 a 2 až po 7 a 8).



Obrázek 3.4: Přenos dat přes rozhraní SPI (převzato z [8])

3.5 Čtečka karet

Na kitu se nachází slot pro karty formátu MMC a SD. Zapojení navíc dovoluje zjistit, zda je karta přítomna nebo zda není chráněna proti zápisu.

3.5.1 Komunikace

Komunikace probíhá přes SPI rozhraní, karta pracuje jako slave zařízení. S kartou se komunikuje podle komunikačního protokolu SD karet (viz [9]), který definuje formát příkazů, odpovědí a postup při přenosu dat. V tabulce 3.2 najdeme přehled použitých příkazů. Tabulka 3.1 ukazuje, jaká je struktura příkazu a v tabulce 3.3 jsou uvedeny druhy tzv. tokenů.

Každý příkaz má pevně danou strukturu, velikost je vždy 6 bajtů. Přenos příkazu začíná `start bitem` (hodnota 0), následuje `transmission bit` (hodnota 1), poté následuje kód, argument a kontrolní součet (CRC7) příkazu, přenos ukončuje `end bit` (hodnota 1).

část	start bit	transmission bit	kód	argument	CRC7	end bit
délka v bitech	1	1	6	32	7	1

Tabulka 3.1: Struktura příkazu

Kód	**Popis**
CMD0	GO_IDLE_STATE – provede reset paměťové karty
CMD1	SEND_OP_COND – slouží pro inicializaci MMC karty
CMD8	SEND_IF_COND – ověřuje, zda je karta schopna pracovat při dostupném napětí
CMD9	SEND_CSD – žádost o poslání obsahu CSD registru
CMD10	SEND_CID – žádost o poslání obsahu CID registru
CMD12	STOP_TRANSMISSION – zastavuje přenost dat
CMD16	SET_BLOCKLEN – nastavuje velikost bloku
CMD17	READ_SINGLE_BLOCK – žádost o zaslání 1 bloku dat
CMD18	READ_MULTIPLE_BLOCK – žádost o zaslání více bloků
CMD23	SET_BLOCK_COUNT – pro MMC karty nastavuje, s kolika bloky se bude pracovat
CMD24	WRITE_BLOCK – zápis jednoho bloku dat
CMD25	WRITE_MULTIPLE_BLOCK – slouží k zápisu více bloků
CMD55	APP_CMD – sděluje kartě, že následující příkaz je aplikačně specifický (ACMD)
CMD58	READ_OCR – žádost o zaslání OCR registru
ACMD23	SET_WR_BLOCK_ERASE_COUNT - příkaz k přípravě na vymazání bloků
ACMD41	APP_SEND_OP_COND - slouží pro inicializaci karty formátu SD

Tabulka 3.2: Přehled kódů používaných příkazů komunikačního protokolu

Datový přenos je řízen pomocí tzv. tokenů. V případě žádosti o čtení (tzn. čtení bloků dat z paměti karty i obsahu OCR, CID a CSD registrů) každý blok dat začíná **Start Block** tokenem a končí 16 bitovým kontrolním součtem. Byla-li žádost o čtení více bloků, ukončuje se přenos zasláním příkazu CMD12.

Při zápisu dat do paměti se rozlišuje, jestli bude probíhat zápis pouze jednoho bloku dat nebo více bloků. V prvním případě se před samotnými daty pošle **Start Block** token shodný s tokenem pro čtení, v druhém případě se vysílá odlišný **Start Block** token a přenos se ukončuje zasláním **Stop Tran** tokenu.

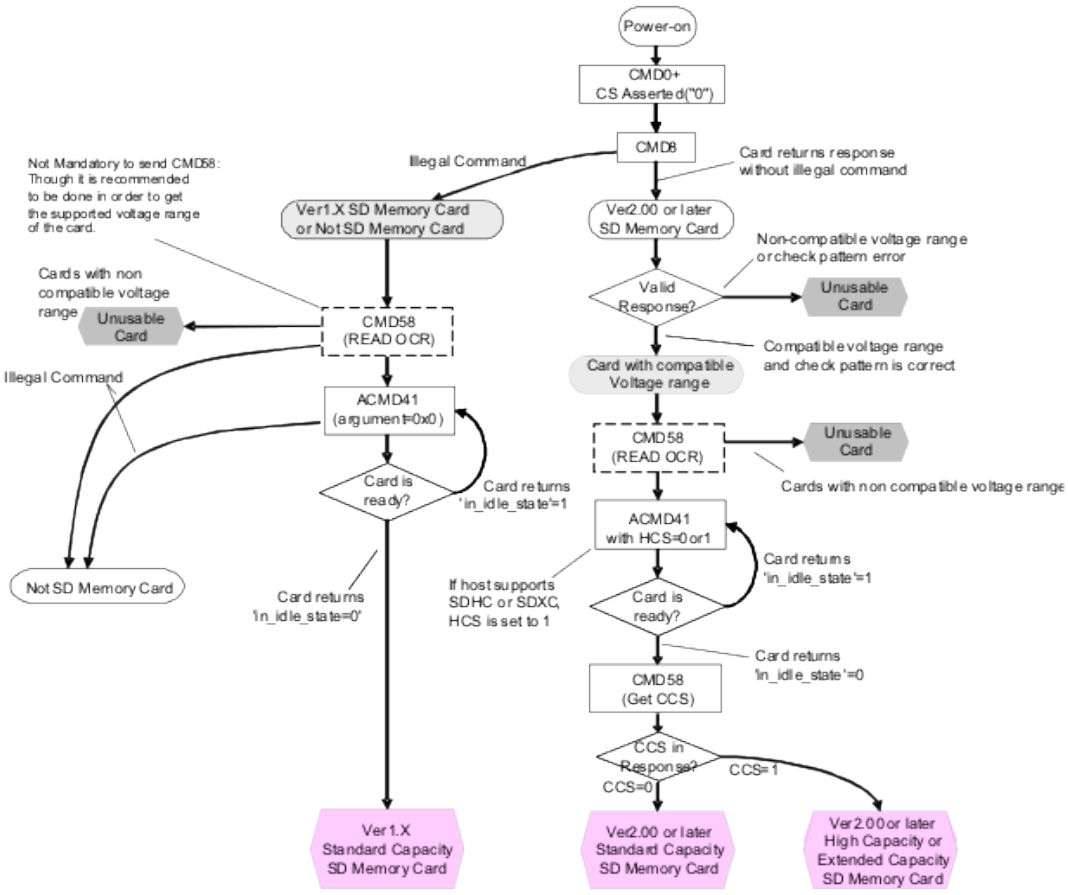
Start Block – čtení, zápis 1 bloku	1	1	1	1	1	1	1	0
Start Block – zápis více bloků	1	1	1	1	1	1	0	0
Stop Tran – konec zápisu	1	1	1	1	1	1	0	1

Tabulka 3.3: Bitová struktura komunikačních tokenů

3.5.2 Inicializace

Při inicializaci je nejprve nutné kartu uvést do režimu SPI komunikace. Což provedeme zasláním příkazu `CMD0`. Z odpovědi poznáme, zda karta přešla do režimu SPI. Pokud ano, následuje inicializační sekvence dle 3.5.

Z průběhu inicializace také zjistíme, jakého typu je vložená karta.



Obrázek 3.5: Postup inicializace karty (převzato z [9])

3.6 Kruhový ovladač

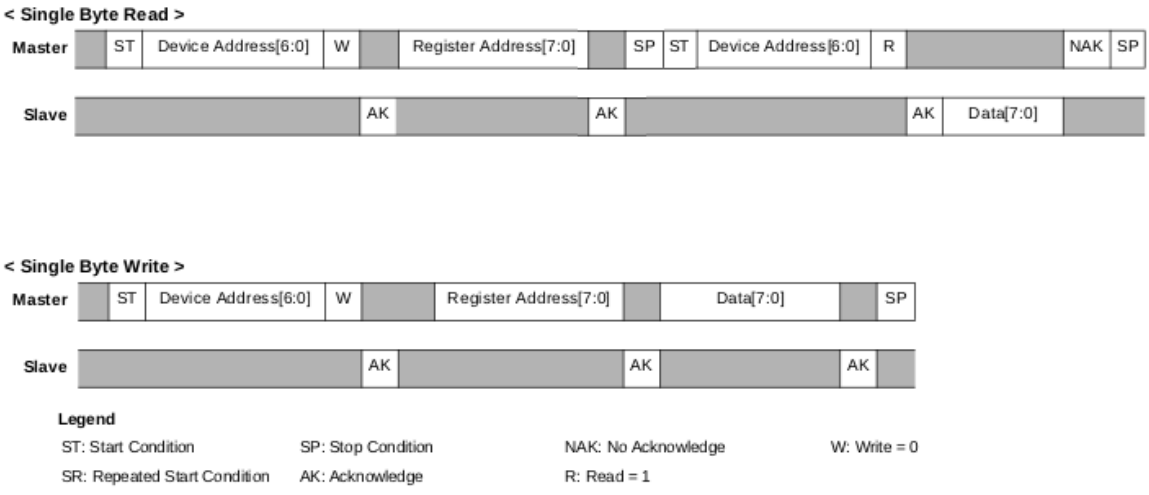
Ovladač je realizován pomocí osmi dotykových plošek, k nimž je připojen kapacitní senzor MPR083[10] z produkce firmy Freescale. Mezi výhody kapacitní technologie patří to, že senzor je schopný reagovat i na pouhé přiblížení. Aby byl detekován stisk, nemusí ani dojít k přímému kontaktu s dotykovou plochou. Integrovaný obvod MPR083 byl navržen pro nahrazení klasických mechanických ovladačů jako jsou přepínače, posuvné či otočné ovladače.

K výhodám řadiče MPR083 patří zásobník až na 30 údajů o stisku nebo uvolnění dotykové plochy, generování přerušení při dotyku/uvolnění dotykové plochy, dále také široká konfigurovatelnost – např. je možné nastavit, jak často se budou vyvolávat přerušení, případně, zda se budou (ne)zaznamenávat údaje o stisku a uvolnění. Obvod sice není schopen snímat současný dotyk na více dotykových plochách, ale to je vzhledem k tomu, že má sloužit jako náhrada jednoduchých mechanických ovladačů, pochopitelné.

3.6.1 Komunikace

Komunikace mikrokontroleru s řadičem MPR083 probíhá přes rozhraní IIC. Komunikace je typu Master-Slave, řídí ji mikrokontroler, MPR083 pouze reaguje na zaslané požadavky (Formáty zpráv pro komunikaci lze vidět v 3.6). Řadič může upozornit mikrokontroler, že jsou pro něj nachystána nějaká data, pouze pomocí IRQ přerušení.

Zápis dat probíhá nastavením modulu IIC do režimu vysílání. Zašleme adresu MPR083 s příznakem zápisu, dále adresu registru, do nějž chceme zapisovat, a hodnotu, kterou do něj chceme vložit. Čtení dat z registru probíhá ve dvou fázích. V první fázi zvolíme registr, což provedeme stejným způsobem jako při zápisu dat, jen neposíláme novou hodnotu registru. V druhé fázi nastavíme modul IIC do režimu vysílání, zašleme adresu MPR083 s příznakem čtení, nastavíme IIC na přijímání dat a čekáme na jejich vyslání.



Obrázek 3.6: Formát zprávy pro čtení a zápis (dle [10])

Módy provozu

Řadič MPR083 může pracovat v režimu `Stop` nebo režimu `Run`. V režimu `Run` probíhá skenování elektrod připojených na dotykové plošky, generují se přerušení a je možné sledovat hodnoty registrů. Režim `Stop` se využívá ve chvíli konfigurace a inicializace řadiče MPR083, v tomto režimu nedochází ke skenování elektrod, nejsou tedy detekovány žádné doteky, avšak kromě čtení hodnoty registrů, je možný i jejich zápis.

3.6.2 Přehled registrů

Pro nastavení a práci s kapacitním senzorem je k dispozici sada registrů:

- `FIFO Register` – Slouží pro čtení ze zásobníku.
- `Fault Register` – Obsahuje informaci, zda není detekován zkrat.
- `Rotary Status Register` – Obsahuje informaci o tom, zda je stisknuta klávesa.
- `Rotary Configuration Register` – Slouží pro nastavení vlastností FIFO registru, povolení automatické kalibrace a zapíná dotykový senzor.
- `Sensitivity Threshold Register` – Slouží pro nastavení citlivosti.
- `Master Tick Period Register` – Nastavuje délku periody hlavních hodin řadiče.
- `Configuration Register` – Nastavuje přerušení, mód činnosti.

3.7 Numerická klávesnice

Klávesnice se skládá z 12 dotykových plošek ovládaných řadičem MPR121[11] firmy Freescale. Snímání dotykových plošek je řešeno pomocí kapacitní technologie. Řadič je druhou generací obvodů pro snímání dotyků na kapacitní bázi. Na rozdíl od řadiče MPR083 disponuje 12 elektrodami pro připojení dotykových plošek. Navíc má senzor, který detekuje přiblížení k dotykovým plochám na vzdálenost několika centimetrů. Oproti MPR083 je také možná detekce současného stisknu na více dotykových plochách.

Rovněž možnosti konfigurace jsou daleko větší. Řadič umožňuje nastavovat citlivost snímání pro jednotlivé elektrody, případně některé elektrody úplně vypnout.

Integrovaný obvod MPR121 obsahuje AD převodník s 10 bitovým rozlišením, který se používá pro analýzu naměřených hodnot. Díky ní se odstraňuje rušení způsobené okolním prostředím a zdokonaluje se systém detekce dotyku.

Obvody s MPR121 se často používají v zařízeních, která používají více tlačítek – např. dálková ovládání, mobilní telefony, MP3 přehrávače apod.

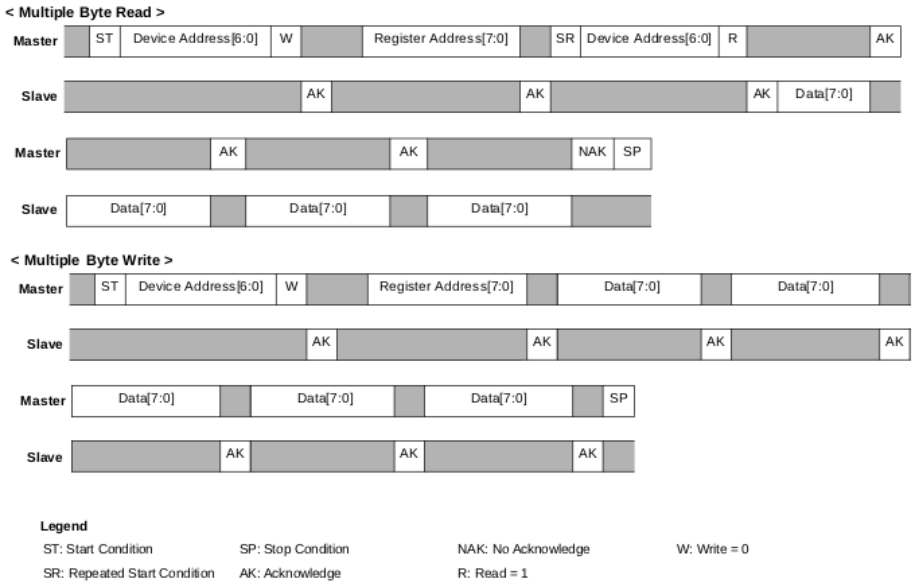
3.7.1 Popis komunikace

Komunikace probíhá přes rozhraní IIC, je řízena mikrokontrolerem. Integrovaný obvod MPR121 funguje jako slave zařízení, komunikuje pouze, pokud je o to požádán. Vyjímku tvoří situace, kdy je detekován stisk dotykové plošky – v tu chvíli řadič generuje IRQ přerušení a je na mikrokontroleru, aby provedl jeho obsluhu.

Formát komunikace je obdobný jako u MPR083. Chceme-li poslat data, nastavíme IIC modul mikrokontroleru do režimu vysílání a vyšleme přes něj adresu MPR121 s příznakem zápisu, následně pošleme adresu registru, do nějž chceme zapisovat a posíláme data.

Chceme-li nastavit více registrů najednou, nemusíme pokaždé inicializovat novou komunikaci. Ve chvíli, kdy řadič obdrží a zapíše hodnotu do zvoleného registru, automaticky adresu inkrementuje a jakmile obdrží další data, zapíše je na novou adresu.

Situace se čtením probíhá ve dvou fázích, nejprve se provede zápis, ve kterém se zvolí adresa registru, následně se znovu inicializuje komunikace, ale tentokrát se s adresou řadiče posílá příznak čtení, poté se IIC modul nastaví do režimu přijímání dat a čtení je zahájeno. Průběh komunikace vidíme na obrázku 3.7.1.



Obrázek 3.7: Formát zprávy pro čtení a zápis (převzato z [11])

Inicializace

Stejně jako obvod MPR083 je nutno i MPR121 před zahájením činnosti řádně nastavit. Nastavování registrů je možné pouze v režimu `Stop` – v tom se řadič nachází po zapnutí. Následně je nutné konfigurovat registry jednotlivých elektrod na požadovanou hodnotu. Dále se nastavují registry systému filtrování dat. Jako poslední je zapsán tzv. `Electrode Configuration Register`, který povoluje činnost jednotlivých elektrod. Poté se MPR121 nachází v `Run` módu.

Kapitola 4

Knihovna

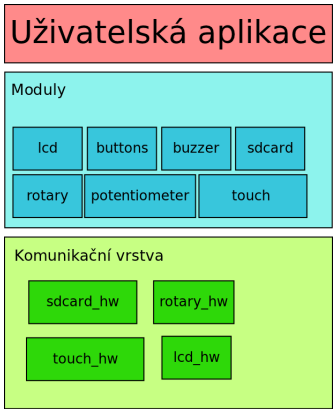
Při navrhování knihovny je důležité dbát na to, aby programátorovi pomáhala s tvorbou aplikace, nikoli aby mu práci komplikovala. Snažil jsem se o to, aby knihovna byla snadno použitelná, přehledná a nenáročná na paměť.

4.1 Struktura

Rozhodl jsem se pro rozdělení knihovny na řadu na sobě nezávislých modulů, z nichž každý pokrývá právě jednu periferii. Pro periferie, u kterých dochází ke komunikaci mezi mikrokontrolerem na kitu a čipem v periferii (například kapacitní senzory, LCD displej), je oddělena aplikační vrstva od vrstvy komunikační tak, že je pro ni vytvořen zvláštní modul. Uživatel má k dispozici funkce pro obsluhu řadiče periferie a komunikace na úrovni modulů na čipu je mu skryta. Vše je patrnější z 4.1.

Díky tomuto rozdělení je snadné nahradit komunikační úroveň za jinou. Příkladem může být řadič LCD displeje, který je uzpůsoben pro komunikaci nejen přes rozhraní SPI, ale i přes paralelní rozhraní. Na kitu je řadič LCD připojen přes SPI, pokud bychom jej chtěli v jiné aplikaci připojit přes paralelní rozhraní, stačí pouze vyměnit komunikační modul.

Chování jednotlivých periferií je nastavitelné pomocí konfiguračních souborů.



Obrázek 4.1: Struktura knihovny

4.2 Implementace

Při vytváření zdrojových kódů jsem vycházel z katalogového listu k mikrokontroleru MC9S08JM60[1] a z refererenčního manuálu mikrokontroleru MCF51JM128[2].
Dalším významným zdrojem informací mi byla kniha o programování mikrokontrolerů rodiny HCS08 [12].

4.2.1 Programovací jazyk

Z programovacích jazyků jsem si mohl vybrat mezi asemblerem a jazykem C. Při výběru jsem rozhodoval na základě těchto vlastností:

- Čitelnost kódu – Jazyk C je oproti asembleru výrazně čitelnější.
- Efektivita kódu – Asemblerovský kód je efektivnější.
- Přenositelnost kódu – Knihovna má být použitelná na architektuře 8 bitového jádra HCS08 i 32 bitového jádra ColdFire. Tuto kompatibilitu umožňuje jazyk C.

Ve výsledku jsem se rozhodl pro použití jazyka C.

4.2.2 Vývojové prostředí

Pro vývoj aplikací pro mikrokontrolery je výhodné používat vývojové prostředí, jelikož programátorovi pomáhá např. s doplňováním kódu, hlídáním závislostí a zejména s odhalováním případných chyb. Firma Freescale pro mikrokontrolery dodává integrované vývojové prostředí CodeWarrior for Microcontrollers v6.3, ve kterém jsem knihovnu vytvářel.

Toto prostředí kromě editoru zdrojových textů zahrnuje i kompilátor a debugger komunikující s mikrokontrolerem. Dále je zde integrován nástroj Processor Expert, který usnadňuje nastavování rozličných vlastností mikrokontroleru – pro konkrétní mikrokontroler může nastavit frekvenci sběrnice, konfigurovat moduly i hodnotu a směr na jednotlivých pinech.

V případě, že chceme mikrokontroler programovat pomocí USBDM programátoru, je nutné nainstalovat ovladače pro USBDM a nástroje pro integraci USBDM do vývojového prostředí. Tyto nástroje je možno nalézt na [3].

4.3 Aplikační rozhraní

4.3.1 Tlačítka

Modul pro obsluhu pětice tlačítek se skládá z konfiguračního souboru `buttons_config.h`, hlavičkového souboru `buttons.h` a souboru `buttons.c`.

Konfigurace

V konfiguračním souboru nalezneme makra, která mapují jednotlivá tlačítka na konkrétní piny portů:

- `BUTTON_XXX_PIN` – Číslo pinu, na který je tlačítko `XXX` (kde `XXX` je `UP`, `DOWN`, `LEFT`, `RIGHT` nebo `OK`) připojeno.
- `BUTTON_XXX_PORT` – Port, na kterém se nachází pin tlačítka.
- `BUTTON_XXX_KBI_PIN` – Určuje, který pin KBI modulu tlačítku přísluší.
- `BUTTON_XXX_LED` – Číslo pinu, na nějž je připojeno podsvícení.
- `BUTTON_XXX_LED_PORT` – Port, který patří podsvícení.

Dále také makra ovlivňující chování modulu:

- `BUTTONS_KBI_ID` – Hodnota tohoto makra nastavuje, který modul KBI přerušení se používá.
- `BUTTONS_EDGE` – Hodnota tohoto makra určuje, zda KBI modul vyvolá přerušení na základě sestupné hrany (hodnota 0), nebo vzestupné hrany (hodnota 1).
- `BUTTONS_USE` – Uživatel zvolí, která z tlačítek bude aplikace používat.
- `BUTTONS_USE_INTERRUPT` – Určuje, jestli KBI modul bude vytvářet požadavky na obsluhu přerušení.
- `BUTTONS_USER_HANDLE_INTERRUPT` – Hodnota 0 nastaví obsluhu modulu tak, že knihovna obsluhuje přerušení a na základě stisknutého tlačítka vyvolá událost, pro kterou uživatel musí vytvořit obslužnou funkci. Hodnota 1 znamená, že se o celou obsluhu přerušení postará uživatel.

Funkce modulu

Uživatel následně používá funkce deklarované v hlavičkovém souboru `buttons.h`:

- `void buttons_init()` – Provede inicializaci modulu, na základě konfigurace uvedené v konfigurační souboru nastaví používaná tlačítka a generování přerušení.
- `void buttons_on(byte mask)` – Na základě hodnoty masky nastaví tlačítka, povolí jejich snímání prostřednictvím KBI modulu, zapne podsvícení.
- `void buttons_on(byte mask)` – Funkce dle hodnoty masky vypne příslušná tlačítka.
- `byte buttons_get_state(byte mask)` – Funkce slouží ke zjistění, která tlačítka jsou stisknuta.

- `void buttons_clear_interrupt()` – Odstraňuje příznak přerušení.
- `void buttons_XXX_handler()` – Prototyp funkce (kde `XXX` je `up`, `down`, `left`, `right` nebo `ok`), která obsluhuje událost vyvolanou při stisku příslušného tlačítka. Jestliže uživatel nastavil, že přerušení budou obsluhována knihovnou, musí danou funkci vytvořit.

4.3.2 Potenciometr

Modul pro obsluhu potenciometru je tvořen soubory `potentiometer_config.h`, `potentiometer.h` a `potentiometer.c`.

Konfigurace

V konfiguračním souboru jsou definována makra, která slouží k nastavení obsluhy AD převodníku pro měření hodnoty nastavené potenciometrem:

- `POTENTIOMETER_CHANNEL_VAL` – Definuje, na kterém kanálu AD převodníku se nachází potenciometr (na kitu je umístěn na 8. kanálu).
- `POTENTIOMETER_MODE_VAL` – Určí mód převodu, který může být 8, 10 nebo 12 bitový.
- `POTENTIOMETER_CLOCK_VAL` – Nastavuje zdroj hodinového signálu AD převodníku.
- `POTENTIOMETER_CLOCK_DIV_VAL` – Určuje hodnotu předděličky hodinového signálu.
- `POTENTIOMETER_ADCO_VAL` – Nastavuje, zda po skončení konverze okamžitě začne konverze nová.
- `POTENTIOMETER_USE_INTERRUPT` – Určuje, jestli budou generovány požadavky na obsluhu přerušení.
- `POTENTIOMETER_USER_HANDLE_INTERRUPT` – Definuje, zda se o obsluhu přerušení stará knihovna, nebo zda je v režii uživatele.

Funkce

Modul prostřednictvím hlavičkového souboru `potentiometer.h` uživateli nabízí následující funkce:

- `word potentiometer_get_state()` – Slouží pro zjištění naměřené hodnoty. V případě, že AD převodník není nastaven pro generování přerušení, zahajuje převod a čeká na jeho dokončení.
- `void potentiometer_start()` – V případě, že je modul nastaven pro vytváření přerušení, tato funkce spouští měření hodnoty potenciometru. Jakmile je převod dokončen, vyvolá se přerušení.
- `void potentiometer_handler()` – Prototyp funkce, která se provede po skončení převodu. Tuto funkci musí uživatel vytvořit, definoval-li, že ponechá obsluhu přerušení na knihovně.

4.3.3 Piezo reproduktor

Část knihovny věnující se obsluze piezo reproduktoru se skládá ze souborů `buzzer.h`, `buzzer.c` a konfiguračního souboru `buzzer_config.h`.

Konfigurace

V konfiguračním souboru nalezneme makra, která je nutná nastavit před použitím modulu:

- `BUZZER_BUSCLK` – hodnota frekvence vnitřní sběrnice
- `BUZZER_PRESCALER` – hodnota předděličky
- `BUZZER_CLK_FREQUENCY` – Frekvence hodin modulu časovače, slouží pro výpočet generované frekvence. Je rovna $\texttt{BUZZER\_BUSCLK} \div 2^{\texttt{BUZZER\_PRESCALER}}$.
- `BUZZER_USER_HANDLE_INTERRUPT` – Rozhoduje o tom, zda se o obsluhu přerušení postará uživatel nebo knihovna.

Funkce

Modul poskytuje jednoduché funkční rozhraní:

- `void buzzer_init()` – Počáteční inicializace modulu.
- `void buzzer_generate_sound(word frequency)` – Slouží pro generování tónu o zvolené frekvenci.
- `void buzzer_keep_frequency()` – Tato funkce musí být umístěna v přerušení generovaném při dosažení hodnoty odpovídající zvolené frekvenci. Znovu nastaví hodnotu pracovního registru čítače, čímž se pokračuje v generování tónu.
- `void buzzer_clear_interrupt()` – Odstraňuje příznak přerušení.

4.3.4 LCD displej

Část knihovny sloužící pro komunikaci s displejem EA DOG-M 162 se sestává ze dvou modulů – jeden se stará o komunikaci mezi řadičem displeje a mikrokontrolerem, druhý obstarává obslužné funkce. Modul pro komunikaci tvoří soubory `lcd_hw.h` a `lcd_hw.c`, modul s definicemi obslužných funkcí se skládá ze souborů `lcd.h` a `lcd.c`. Oba moduly využívají konfiguraci v souboru `lcd_config.h`.

Konfigurace

V konfiguračním souboru nalezneme makra, která slouží k namapování signálů LCD displeje na konkrétní piny.

- `LCD_XXX` – Určuje pozici pinu `XXX`, kde `XXX` je signál `RS` nebo `CS`, či pin barvy podsvícení `RED`, `GREEN` nebo `BLUE`.
- `LCD_XXX_DATA` – port příslušného pinu
- `LCD_XXX_DIR` – registr nastavující směr pinu
- `LCD_SPI_ID` – Slouží pro vybrání SPI modulu, přes který probíhá komunikace s řadičem displeje.
- `LCD_PWM_BACKLIGHT` – Určuje, zda je možno ovládat odstín podsvícení.

Funkce a makra

Modul obstarávající komunikaci s řadičem přes SPI rozhraní definuje následující funkce:

- `void lcd_hw_init(void)` – Inicializuje SPI modul.
- `void lcd_send_command(byte command)` – Slouží pro poslání příkazu řadiči (signál RS má hodnotu 1).
- `void lcd_send_data(byte data)` – Slouží pro zapsání dat do paměti řadiče (signál RS má hodnotu 0).

V modulu s obslužnými funkcemi nalezneme definice funkcí pro práci s displejem:

- `void lcd_init()` – Provede počáteční inicializaci modulu.
- `void lcd_putchar(char c)` – Vypíše znak na displej.
- `int lcd_puts(char *text)` – Vypíše řetězec na displej.
- `void lcd_create_character(byte id, byte *format)` – Slouží pro vytvoření jednoho z 8 znaků, které má uživatel možnost vytvořit.
- `void lcd_display_shift(byte direction)` – Posouvá displejem (1 – doprava, 0 – doleva).
- `void lcd_cursor_shift(byte direction)` – Posouvá kurzorem (1 – doprava, 0 – doleva).
- `void lcd_set_backlight(byte red, byte green, byte blue)` – Nastavuje barvu podsvícení displeje.
- `void lcd_set_contrast(byte contrast)` – Nastavuje kontrast displeje.

4.3.5 Čtečka karet

Pro obsluhu čtečky karet slouží modul zajišťující komunikaci přes SPI rozhraní (`sdcard_hw.h` a `sdcard_hw.c`) a modul s uživatelskými funkcemi (`sdcard.h` a `sdcard.c`). Konfigurace se nastavuje v souboru `sdcard_config.h`.

Konfigurace

Konfigurační soubor slouží pro namapování signálů SD karty na konkrétní piny mikrokontroleru.

- `SDCARD_XXX` – Určuje pozici pinu `XXX`, kde `XXX` je `CS` – signál „chip select“, `DET` – pin pro detekci karty, `WP` – pin pro zjištění, zda je na kartě zapnuta ochrana proti zápisu.
- `SDCARD_XXX_DATA` – Datový port příslušného pinu.
- `SDCARD_XXX_DIR` – Směrový port příslušného pinu.
- `SDCARD_SPI_ID` – Určuje, na jaký modul SPI je připojena čtečka karet.
- `SDCARD_DETECTION` – Nastavuje, zda se používá detekce karty.
- `SDCARD_WRITE_PROTECTION` – Nastavuje, zda se ověřuje, zda je karta chráněna proti zápisu.
- `SDCARD_BUSCLK` – Frekvence vnitřní sběrnice.

Datové typy

Aby byla práce s daty poskytovanými kartou snazší, modul definuje následující datové typy:

- `OCR_REG` (`Operation Condition Register`) – Z tohoto 32 bitového registru registru může uživatel zjistit, při jakém napětí může karta pracovat a zda se jedná o kartu vysokokapacitního formátu.
- `CID_REG` (`Card IDentification`) – Jedná se o 16 bajtový registr obsahující informace o výrobci, datu vyrobení karty a sériovém číslu karty.
- `CSD_REG` (`Card-Specific Data`) – Jak je z názvu patrné, tento 16 bajtový registr obsahuje rozličné údaje o kartě – např. o velikosti karty, jaká je maximální velikost bloku pro zápis a čtení, jaká je maximální rychlost přenosu dat apod.

Funkce a makra

Modul pro komunikaci přes SPI rozhraní definuje následující funkce:

- `void sdcard_hw_init(byte baudrate)` – Inicializacuje SPI modul, nastavuje frekvenci hodin SPI.
- `byte sdcard_send_command(byte command, dword argument, byte crc)` – Slouží pro poslání příkazu SD kartě.
- `byte sdcard_get_byte()` – Přečte jeden bajt z datového registru SPI.

- `byte sdcard_get_bytes(byte *buffer, word count)` – Uskuteční datový přenos z SD karty.

Modul přístupný uživateli definuje datové typy odpovídající struktuře registrů SD karty a makra pro práci s nimi, dále také zajišťuje funkce pro přenos dat.

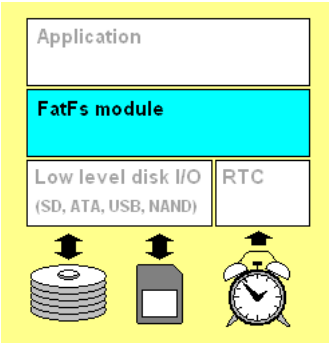
- `byte sdcard_init(void)` – Provede inicializaci karty.
- `byte sdcard_get_type(void)` – Zjistí, jaký je typ vložené karty.
- `byte sdcard_read_blocks(byte *buffer, dword sector_number, byte count)` – Funkce slouží pro získání bloků dat z karty, čtení začíná od uvedeného sektoru.
- `byte sdcard_write_blocks(const byte *buffer, dword sector_number, byte count)` – Funkce slouží pro zápis bloku dat na kartu, zápis probíhá od uvedeného sektoru.
- `byte sdcard_get_ocr(OCR_REG *reg)` – Zjistí obsah `OCR` registru.
- `byte sdcard_get_cid(CID_REG *reg)` – Slouží pro získání obsahu `CID` registru.
- `byte sdcard_get_csd(CSD_REG *reg)` – Slouží pro zjištění obsahu `CSD` registru.

Práce se souborovým systémem

Aby byla práce s daty na SD kartě možná, je nezbytné mít funkce pro práci s jejím souborovým systémem.

Pro tento účel jsem se rozhodl použít modul `FatFs`[13]. `FatFs` je knihovna funkcí pro obsluhu souborového systému formátu `FAT` určená pro použití ve vestavěných systémech. Knihovna není závislá na nosiči dat, může jím být pevný disk, flash disk nebo v našem případě paměťová karta. Modul obsahuje funkce pro vytváření a rušení souborů a adresářů, funkce pro čtení i zapisování obsahu souborů.

Aplikační rozhraní knihovny je odděleno od vrstvy komunikující s datovým médiem (viz 4.2). Pro zprovoznění `FatFs` bylo nutné tuto komunikační vrstvu vytvořit, k tomu jsem využil vytvořený modul pro práci s SD kartou.



Obrázek 4.2: Struktura knihovny FatFS, převzato z [13]

4.3.6 Kruhový ovladač

Část knihovny pro práci s kruhovým ovladačem se skládá z modulu implementujícího komunikaci s řadičem MPR083 přes rozhraní IIC a modulu (`rotary_hw.h` a `rotary_hw.c`) a z modulu s funkcemi pro práci s ovladačem (`rotary.h` a `rotary.c`). Konfigurace se nachází v souboru `rotary_config.h`.

Konfigurace

V konfigurační souboru se nastavují vlastnosti modulu.

- `ROTARY_ADDR` – IIC adresa MPR083, je-li adresní pin řadiče uzemněn, je IIC adresa `0x4C`, jinak `0x4D`. Na kitu je první zapojení.
- `ROTARY_IIC_ID` – Hodnota určuje, na který modul IIC je MPR083 připojeno.
- `ROTARY_USE_INTERRUPT` – Nastavuje, zda se generují přerušení.
- `ROTARY_USER_HANDLE_INTERRUPT` – Rozhoduje, zda se o obsluhu přerušení stará knihovna nebo uživatel.
- `ROTARY_USE` – Slouží pro nastavení používaných tlačítek.
- `ROTARY_LOG_TOUCH` – Nastavuje, zda se do FIFO zásobníku budou zaznamenávat detekce stisku.
- `ROTARY_LOG_RELEASE` – Nastavuje, zda do FIFO zásobníku budou ukládány informace o uvolnění dotykové plochy.
- `ROTARY_SENSITIVITY_VAL` – Nastavuje hodnotu prahu citlivosti snímámí, větší hodnota znamená menší citlivost.
- `ROTARY_MTP_VAL` – Nastavuje délku periody primárních hodin řadiče.
- `ROTARY_TASP_VAL` – Určuje, jak často bude probíhat kontrola, jestli nenastala změna na dotekové plošce.

Datové typy

K usnadnění práce s FIFO registrem jsem vytvořil abstraktní datový typ `ROTARY_FIFO_KEY`, který svou strukturou odpovídá skutečné podobě FIFO registru. Pro usnadnění práce s tímto typem, byla vytvořena následující makra:

- `ROTARY_FIFO_MORE_DATA(reg)` – Dává informaci, zda v FIFO registru budou data i v následujícím čtení.
- `ROTARY_FIFO_EMPTY(reg)` – Registr neobsahuje žádná data.
- `ROTARY_FIFO_OVERFLOW(reg)` – Došlo k přetečení registru, nejaktuálnější data jsou ztracena.
- `ROTARY_FIFO_TOUCHED(reg)` – Určuje, zda došlo ke stisku nebo uvolnění dotykové plošky.
- `ROTARY_FIFO_GET_KEY(reg)` – Vrací hodnotu masky pro stisknutou plošku.

Funkce, makra

Modul vrstvy komunikace poskytuje následující funkce:

- `void rotary_hw_init()` – Provádí počáteční nastavení IIC modulu.
- `void rotary_set_register(byte reg_addr, byte value)` – Slouží pro nastavení hodnoty zvoleného registru.
- `byte rotary_get_register(byte reg_addr)` – Funkce slouží pro získání hodnoty zvoleného registru.

Modul přístupný uživateli definuje funkce:

- `void rotary_init()` – Inicializuje MPR083 dle hodnot nastavených v konfiguračním souboru.
- `byte rotary_get_state()` – Vrací hodnotu masky pro aktuálně stisknutou dotykovou plošku.
- `ROTARY_FIFO_KEY rotary_get_first()` – Vrací obsah FIFO registru.
- `byte rotary_error()` – Vrací obsah chybového registru.
- `void rotary_clear_interrupt()` – Odstraňuje příznak přerušení.
- `void rotary_clear_buffer()` – Vyprázdní FIFO registr.
- `void rotary_error_handler(byte error)` – Prototyp funkce pro obsluhu přerušení vyvolaného chybou, parametr určuje, o jakou chybu se jedná. Definoval-li uživatel, že se o obsluhu přerušení stará knihovna, musí tuto funkci vytvořit.
- `void rotary_key_XXX_handler(byte touched)` – Prototyp funkce pro obsluhu události vyvolané stiskem či uvolněním dotykové plošky (`XXX` odpovídá jedné z hodnot `1` až `8`). Stará-li se o obsluhu přerušení knihovna, uživatel musí funkci vytvořit.

4.3.7 Numerická klávesnice

Část pro obsluhu numerické klávesnice se skládá z modulu pro komunikaci s řadičem klávesnice (`touch_hw.h` a `touch_hw.c`) a z modulu obslužných funkcí (`touch.h` a `touch.c`).

Konfigurace

- `TOUCH_ADDR` – IIC adresa řadiče MPR121. Hodnota adresy je v intervalu `0x5A` až `0x5D` v závislosti na tom, zda je na adresní pin řadiče přivedena nízká úroveň, vysoká úroveň, signál IIC hodin nebo datový vodič.
- `TOUCH_IIC_ID` – Volba IIC modulu.
- `TOUCH_USE` – Určuje, které klávesy jsou používány.
- `TOUCH_THRESHOLD` – Nastavuje práh citlivosti pro detekci doteku.
- `RELEASE_THRESHOLD` – Nastavuje hodnotu, po níž dojde k detekci uvolnění klávesy.
- `TOUCH_USE_INTERRUPT` – Určuje, zda se generují přerušení.
- `TOUCH_USER_HANDLE_INTERRUPT` – Určuje, zda se o obsluhu přerušení stará knihovna nebo uživatel.

Funkce

Funkce modulu pro IIC komunikaci.

- `void touch_hw_init()` – Provede inicializaci IIC modulu.
- `void touch_send_bytes(byte *data, byte n)` – Slouží pro poslání dat do řadiče MPR121.
- `byte touch_read_byte(byte reg_addr)` – Vrátí obsah zvoleného registru.

Funkce přístupné uživateli.

- `void touch_init()` – Provede inicializaci a nastaví modul dle konfiguračního souboru.
- `word touch_get_state()` – Vrátí masku stisknutých kláves.
- `void touch_clear_interrupt()` – Slouží pro odstranění příznaku přerušení.
- `void touch_key_XXX_handler()` – Prototyp obslužné funkce (`XXX` odpovídá jedné z hodnot `0` až `9`, nebo `hash`, `star`, či `proximity`). Je-li obsluha přerušení v režii knihovny, uživatel musí definovat tuto obslužnou funkci.

Kapitola 5

Grafická knihovna Freescale eGUI

Pro výukový kit byl vytvořen přídavný modul s dotykovým displejem. Aby bylo možné využívat jeho funkcí, je nezbytné mít k dispozici grafickou knihovnu.

Firma Freescale, jejímiž mikrokontrolery je kit osazen, nabízí ke stažení grafickou knihovnu Freescale eGUI (D4D) [14]. Aby bylo možné knihovnu na kitu používat, musel jsem vytvořit řadič pro dotykový displej přídavného modulu.

5.1 Vlastnosti knihovny

Grafická knihovna Freescale eGUI umožňuje tvorbu grafického uživatelského rozhraní na mikrokontrolerech. Primárně je určena pro 32 bitové mikrokontrolery, ale je možno ji provozovat i na mikrokontrolerech 8 bitových. Mezi další její vlastnosti patří:

- Podpora různých rozlišení displeje
- Zobrazení 65 536 různých barev (formát RGB565)
- Nezávislost na platformě – snadná přenositelnost knihovny mezi architekturami
- Objektový styl – obsahuje sadu základních objektů, které spolu komunikují zasíláním zpráv.
- Pečlivě zpracovaná dokumentace

5.1.1 Písma a obrázky

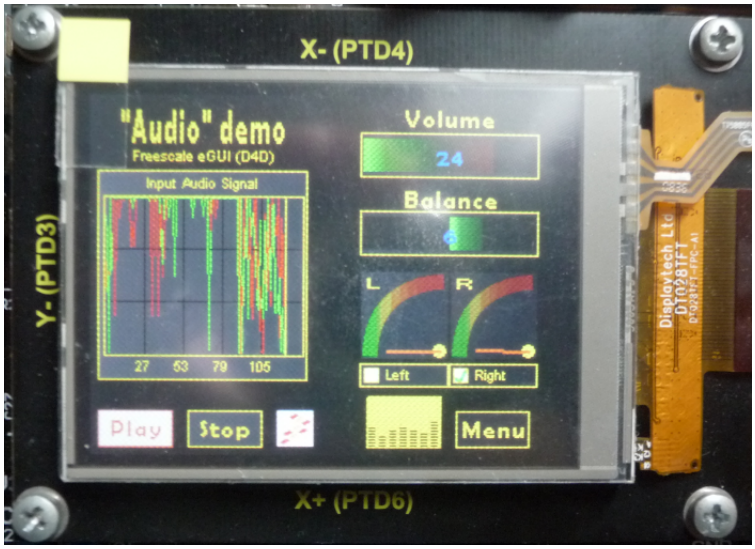
Ke knihovně je dodáván nástroj Freescale Embedded GUI Converter Utility 2.0, který slouží pro převod fontů a obrázků do dat v jazyce ANSI C, ve formátu využitelného knihovnou. Výsledná data jsou uložena ve zvláštním hlavičkovém a zdrojovém souboru, který tak můžeme používat nezávisle na aplikaci.

Program obsahuje nástroje pro úpravu obrázků (změna palety barev, rotace, změna velikosti) a nástroje pro úpravu podoby jednotlivých znaků.

5.1.2 Práce s grafickou knihovnou

Knihovna pro práci nabízí řadu objektů. Mezi ty základní patří:

- `D4D_SCREEN` – Okno je základním stavebním kamenem každé aplikace. Chová se jako „kontejner“ dalších objektů.
- `D4D_BUTTON` – Tlačítko funguje tak, jak bychom od objektu podobného typu očekávali. Má nastavitelný text, barvu. Je možné jej barevně odlišit, pokud je aktivní apod.
- `D4D_CHECKBOX` – Objekt slouží jako klasické zatrhávací políčko.
- `D4D_GAUGE` – Ručičkový ukazatel má využití např. pro zobrazování teploty, rychlosti.
- `D4D_PICTURE` – Slouží pro manipulaci s bitmapovým obrázkem.
- `D4D_SLIDER` – Posuvník, používá se např. jako ukazatel stavu nebo pro nastavování hodnoty.
- `D4D_MENU` – Objekt představuje klasické menu s několika položkami.
- `D4D_GRAPH` – Objekt slouží pro vizualizaci hodnot do grafové podoby.
- `D4D_CONSOLE` – Konzole poskytuje prostředky pro psaní textu, včetně možnosti manipulování s kurzorem apod.
- `D4D_TEXTBOX` – Objekt představující textové pole.

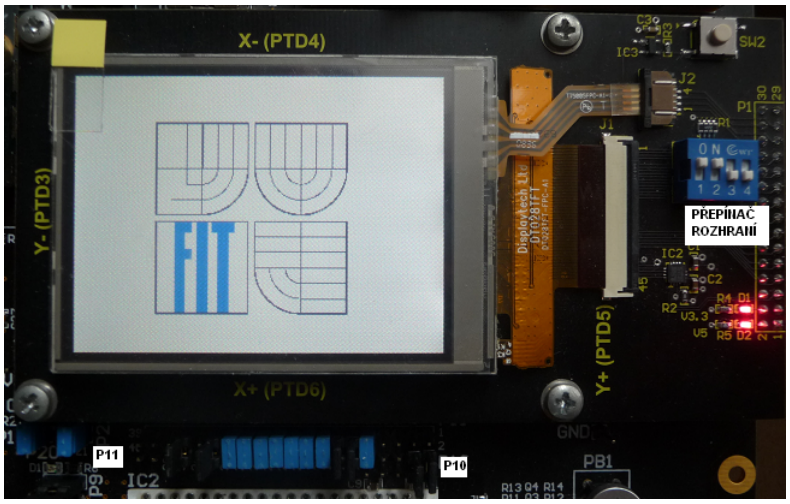


Obrázek 5.1: Objekty grafické knihovny eGUI

5.2 Přídavný modul s dotykovým displejem

Jedná se o dotykový TFT displej s rozlišením 240×320 pixelů a úhlopříčkou $2,8''$, umožňuje zobrazit 262 144 různých barev (formát RGB666).

Informace o modulu jsem získal z [15], displej je řízen řadičem ILI9320[16].



Obrázek 5.2: Přídavný modul

5.2.1 Dotykové rozhraní

Na povrchu displeje je dotyková vrstva, která je založena na 4 vodičové rezistivní technologii. Ta pracuje na principu dvou oddělených odporových vodivých vrstev, ke každé z nich jsou připojeny dva vodiče. Při doteku dojde ke spojení vrstev, vzniká elektrický obvod a místo doteku funguje jako odporový dělič. Pomocí AD převodníku vypočteme hodnotu napětí a z ní odvodíme místo dotyku.

5.2.2 Komunikace

Komunikace s řadičem displeje může probíhat přes rozhraní SPI, přes paralelní rozhraní i80, rozhraní VSYNC a RGB. Poslední dvě rozhraní jsou vhodná pro zobrazování pohyblivého obrazu, využívají pokročilé techniky přímého přístupu do paměti, aby eliminovaly prodlevy při přenosu dat. Naproti tomu rozhraní i80 a SPI přistupuje do paměti pomocí speciálních registrů.

Grafická knihovna obsahuje ovladače rozhraní i80 a SPI v 8 bitových verzích, další informace se budou vztahovat k těmto rozhraním.

Pro komunikaci se využívají tři registry řadiče:

- `Index Register` (IR) – Jedná se o 16 bitový registr, který specifikuje adresu řídícího registru řadiče, s nímž bude probíhat přenos dat (čtení nebo zápis).
- `Write-Data Register` (WDR) – Tento 18 bitový registr slouží pro dočasné uložení dat, která budou zapsána do řídícího registru, či paměti GRAM.
- `Read-Data Register` (RDR) – Jedná se o 18 bitový registr, do nějž se dočasně ukládají data z paměti GRAM.

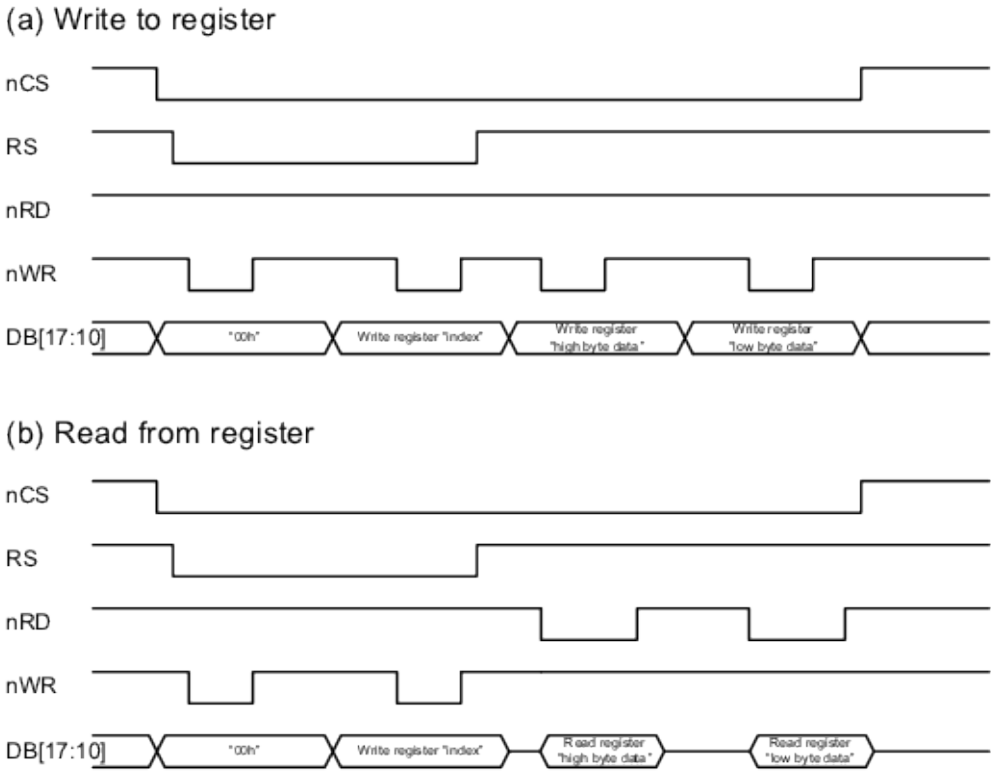
Přenos barev

Zápis dat do grafické paměti prostřednictvím 18 bitového WDR registru je možný jedním ze tří způsobů:

- První používá tři přenosy, z nichž každý obsahuje jednu barevnou složku – formát barev RGB666.
- I druhá varianta používá tři přenosy, avšak v prvním přenosu se přenáší pouze dva bity, v následujících dvou převodech je přeneseno zbývajících 16 bitů – formát RGB666.
- Poslední možností je použít pouze dvou přenosů – formát RGB565. Grafická knihovna používá právě tento formát barev, proto je vhodné zvolit tento způsob přenosu, navíc je oproti předchozím rychlejší.

Rozhraní i80

Rozhraní i80 je na kitu přístupné v 8 bitové verzi. Jeho předností je rychlost. Na druhou stranu tím, že není přenášen synchronizační signál, může docházet k chybné interpretaci přenášených dat. Aby se tato možnost eliminovala, je vhodné občas použít speciální synchronizační přenos. Pro nastavení tohoto rozhraní je nutné propojit piny 1 a 2 portu P10 s piny 1 a 3 portu P11, propojit dvojice pinů 9-10 až 29-30 a 31-32 na P11. Dále je potřeba zvolit na přepínači na modulu mód i80/8 bitů. Umístění pinů a přepínače je vidět na obrázku 5.2.



Obrázek 5.3: Formát komunikace přes rozhraní i80 (převzato z [16])

Rozhraní SPI

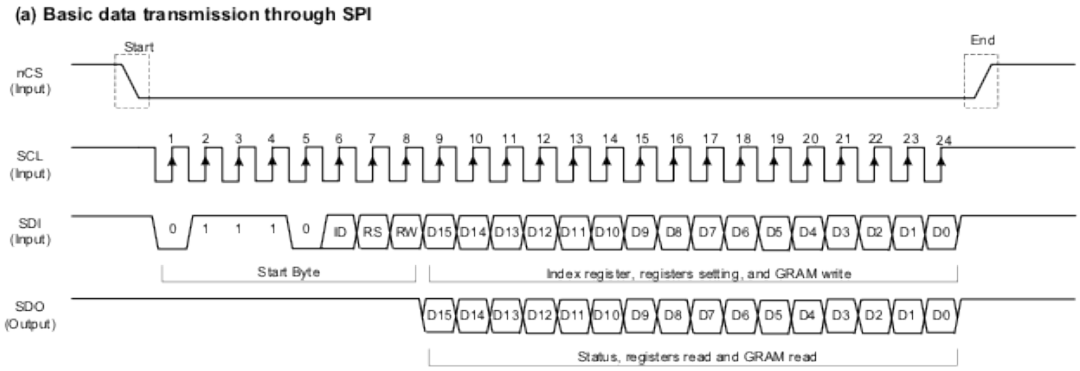
Komunikace přes rozhraní SPI se vyznačuje jednoduchostí. Na kitu stačí propojit piny portu P10 s P11 a na přepínači na modulu zvolit mód SPI, o přenos dat se postará modul SPI na mikrokontroleru.

Komunikace začíná přenesením tzv. `start bajtu`, dle něhož řadič displeje pozná, s jakým z registrů se bude pracovat. Po té následuje přenos samotných dat. Formát komunikace je znázorněn na 5.4.

Nejvyšších pět bitů `start bajtu` má vždy stejnou hodnotu (01110_2), hodnota bitu ID se odvíjí z hodnoty přepínače rozhraní. Bit RS (Register Select) určuje, zda se bude pracovat s index registrem, či s některým z řídících registrů. A nakonec bitem RW se nastavuje, zda bude probíhat čtení nebo zápis.

bit	7	6	5	4	3	2	1	0
hodnota	0	1	1	1	0	ID	RS	RW

Tabulka 5.1: Formát `start bajtu`



Obrázek 5.4: Formát komunikace přes rozhraní SPI (převzato z [16])

5.2.3 Funkce

Řadič vyšším vrstvám knihovny poskytuje následující funkce:

- `unsigned char D4DLCD_Init()` – Provede inicializační sekvenci displeje, nastaví komunikační rozhraní, zapne podsvícení displeje.
- `unsigned char D4DLCD_SetWindow(unsigned short x1, unsigned short y1, unsigned short x2, unsigned short y2)` – Slouží pro vymezení hranic oblasti grafické paměti, do nichž se bude zapisovat.
- `unsigned char D4DLCD_SetOrientation(D4DLCD_ORIENTATION new_orientation)` – Nastavuje orientaci displeje, na výběr jsou možnosti orientace na šířku (Landscape, Landscape180) nebo na výšku (Portrait, Portrait180).
- `unsigned char D4DLCD_Send_PixelColor(unsigned short value)` – Nastavuje pixelu zvolenou barvu.

Kapitola 6

Závěr

Záměrem práce bylo vytvořit knihovnu funkcí pro základní obsluhu periferí a demostrovat schopnosti výukového kitu.

Výsledkem práce je vytvořená knihovna, kterou je možno využívat při výuce hardwarových předmětů. Byla zprovozněna grafická knihovna Freescale eGUI, což výrazně posunulo schopnosti rozšiřujícího modulu s dotykovým displejem.

Díky práci jsem si prohloubil znalosti z oblasti programování mikrokontrolerů, včetně jejich praktické aplikace.

Na práci je možné navázat například vytvořením grafického nástroje, který by vytvářel konfigurační soubory pro periferie. Tím by knihovna získala na využitelnosti.

Literatura

[1] Freescale Semiconductor, Inc. *MC9S08JM60: Technical Data Sheet for MC9S08JM60 and MC9S08JM32* [online]. 2009 [cit. 10. listopadu 2011]. Dostupné na: http://cache.freescale.com/files/microcontrollers/doc/data_sheet/MC9S08JM60.pdf.

[2] Freescale Semiconductor, Inc. *MCF51JM128 ColdFire® Integrated Microcontroller Reference Manual* [online]. 2009 [cit. 30. listopadu 2011]. 558 s. Dostupné na: http://cache.freescale.com/files/32bit/doc/ref_manual/MCF51JM128RM.pdf.

[3] O'Donoghue, P. *USBDM* [online]. [cit. 1. dubna 2012]. Dostupné na: http://usbdm.sourceforge.net/.

[4] Šimek, V. *Schéma výukového kitu*. Brno: FIT VUT v Brně.

[5] TDK. *Piezoelectronic Buzzers: PS series* [online]. 2011 [cit. 28. dubna 2012]. 8 s. Katalogový list. Dostupné na: http://www.tdk.co.jp/tefe02/ef532_ps.pdf.

[6] Schwarz, J. *Systém časování*. Brno: FIT VUT v Brně, 2011. Přednáška z předmětu IMP.

[7] ELECTRONIC ASSEMBLY. *EA DOG-M: Datasheet* [online]. 2010 [cit. 27. února 2012]. 8 s. Dostupné na: http://www.lcd-module.de/eng/pdf/doma/dog-me.pdf.

[8] Sitronix Technology Co. *ST7036: Dot Matrix LCD Controller/Driver* [online]. 2003 [cit. 2. dubna 2012]. 72 s. Dostupné na: http://www.lcd-module.de/eng/pdf/zubehoer/st7036.pdf.

[9] SD Group and SD Card Association. *SD Specifications: Part 1, Physical Layer Simplified Specification Version 3.01* [online]. 2010 [cit. 9. dubna 2012]. 153 s. Dostupné na: https://www.sdcard.org/downloads/pls/simplified_specs/Part_1_Physical_Layer_Simplified_Specification_Ver_3.01_Final_100518.pdf.

[10] Freescale Semiconductor, Inc. *MPR083: Proximity Capacitive Touch Sensor Controller* [online]. 2010 [cit. 20. února 2012]. 35 s. Dostupné na: http://cache.freescale.com/files/sensors/doc/data_sheet/MPR083.pdf.

[11] Freescale Semiconductor, Inc. *MPR121: Proximity Capacitive Touch Sensor Controller* [online]. 2011 [cit. 20. února 2012]. 388 s. Dostupné na: http://cache.freescale.com/files/sensors/doc/data_sheet/MPR121.pdf.

[12] Pereira, F. *HCS08 unleashed: Designer´s guide to the HCS08 Microcontrollers*. 2. vyd. Charleston: BookSurge Publishing, 2009. 413 s. ISBN 1-4196-8592-9.

[13] ChaN. *FatFs: Generic FAT File System Module* [online]. 2011 [cit. 11. března 2012]. Dostupné na: http://elm-chan.org/fsw/ff/00index_e.html.

[14] Freescale Semiconductor, Inc. *Freescale Embedded GUI (D4D)* [online]. 2010 [cit. 10. dubna 2012]. 158 s. Dostupné na: http://cache.freescale.com/files/microcontrollers/doc/ref_manual/DRM116.pdf.

[15] Sekanina, P. *Vývojová deska s mikrokontrolérem a displejem touchscreen.* Brno: FIT VUT v Brně, 2010. 46 s. Bakalářská práce.

[16] ILI TECHNOLOGY CORP. *ILI9320: Datasheet.* 2008. 109 s.

Příloha A

Obsah CD

Na přiloženém CD nalezneme:

- Zdrojové kódy knihovny funkcí pro obsluhu periferií
- Zdrojové kódy řadiče displeje pro knihovnu Freescale eGUI
- Ukázkové aplikace pro jednotlivé periferie
- Katalogové listy a manuály periferií
- Text bakalářské práce ve formátu pdf a LaTeX