Zadání diplomové práce

Název diplomové práce:

Návrh a vývoj pokročilé řídící jednotky lineárního motoru pro přesná laboratorní měření v biomechanice.

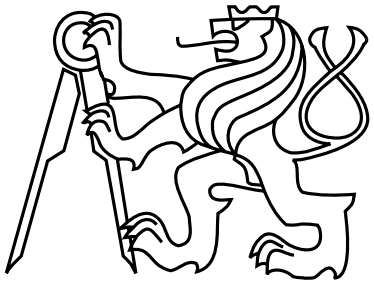
Pokyny pro vypracování:

Vyberte vhodnou platformu pro realizaci řídící jednotky k lineárnímu motoru EZ Limo EZC6E030M–C, který využívá driver ESMC–C2. Řídící jednotka bude podporovat zadané funkce. Pro přesná měření musí řídící jednotka dosahovat vysoké spolehlivosti a rychlosti zpracování příkazů. K řídící jednotce bude dodána demonstrační aplikace a bude navržena pro budoucí použití v systému MATLAB.

Seznam literatury:

Dodá vedoucí práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra číslicového návrhu



Diplomová práce

Návrh a vývoj pokročilé řídící jednotky lineárního motoru pro přesná laboratorní měření v biomechanice

Bc. Matěj Bartík

Vedoucí práce: Dr.-Ing. Martin Novotný

Studijní program: Informatika, strukturovaný, Magisterský

Obor: Projektování číslicových systémů

18. září 2014

Poděkování

Děkuji svému vedoucímu práce a ostatním členům katedry číslicového návrhu ČVUT FIT, dále členům Laboratoře biomechaniky člověka, ČVUT FS za nezměrnou pomoc a rady při všech fázích práce na této diplomové práci.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 7. 5. 2014 .....

Abstract

This diploma thesis describes the design and implementation of control system for linear motor EZ Limo EZC6E030M–C and its control unit ESMC–C2. The control system is designed for precise and reliable laboratory measurements in biomechanics. The control system has been split into software part (control/manipulation application) and hardware part (new advanced control unit implemented on Xilinx's FPGA). The control system and all its parts were exhaustly tested under full operational conditions. The control system actually does not meet EMC directives. New printed circuit board is currently under development to meet the EMC directives.

Abstrakt

Tato diplomová práce se zabývá návrhem a realizací řídícího systému pro lineární motor EZ Limo EZC6E030M–C a jeho řídící jednotku ESMC–C2. Tento systém je navržen pro přesná a spolehlivá měření v biomechanice. Realizace řídícího systému je rozdělena na softwarovou část (obslužná aplikace) a hardwarovou část (nová pokročilá řídící jednotka realizovaná na FPGA firmy Xilinx). Řídící systém a všechny jeho součásti byly pečlivě otestovány za plného provozu. Systém v současné době nesplňuje předpisy týkající se elektromagnetické kompatibility (EMC). Připravuje se nová verze desky plošných spojů, která splní požadavky na elektromagnetickou kompatibilitu.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Nápad k realizaci systému pro řízení lineárního motoru vznikl v roce 2010 v Laboratoři biomechaniky člověka, Ústavu mechaniky, biomechaniky a mechatroniky Fakulty strojní, Českého vysokého učení technického v Praze, kde se zabývají měřením biologických vzorků a jejich mechanických parametrů. Laboratoří pořízený lineární motor EZ Limo EZC6E030M–C s řídící jednotkou ESMC–C2 měl doplnit stávající 3–osou lavici poháněnou trojicí lineárních motorů. To mělo umožnit souběžný provoz některých experimentů a především ušetřit čas a námahu při opakované montáži měřících aparatur a kalibraci řídícího systému pro experimenty, kterým by za normální situace dostačoval lineární motor vykonávají svůj pohyb pouze v jedné ose.

Po zakoupení a otestování motoru a jeho řídící jednotky se zjistila nedostatečná softwarová podpora výrobce pro řídící jednotku, která nedovolovala provádět některé prvky požadované funkcionality. Jako hlavní nedostatek se ukázalo omezení délky prováděného programu, který nedostačoval na provedení některých složitějších pohybů. Tím nebylo možné simulovat pohyby napodobující mechaniku člověka.

Proto bylo rozhodnuto pokusit se vyvinout řídící jednotku vlastní konstrukce, která by splňovala po stránce funkčnosti a spolehlivosti plnohodnotnou náhradu za původní řídící jednotku ESMC–C2. Požadavky na funkcionalitu byly sepsány do externího dokumentu s podrobnými komentáři a příklady použití.

Protože hlavním podpůrným systémem pro výpočty a vizualizaci na Fakultě strojní je systém MATLAB, bude řídící systém realizovaný v systému MATLAB, který bude komunikovat s částí řídícího systému realizovanou v hardwaru. Hardwarová část přímo ovládá lineární motor.

Práce je rozdělena do těchto kapitol: Kapitola dvě rozvádí zadání práce a popisuje úkoly, které z něj vyplývají. Třetí kapitola obsahuje analýzu a návrh řešení všech částí, které je třeba realizovat. Samotná realizace je obsahem čtvrté kapitoly. Pátá kapitola popisuje postupy testování funkce a výkonu realizovaného návrhu. Poslední kapitolu tvoří závěr.

Kapitola 2

Popis problému, specifikace cíle

Hlavním důvodem pro vývoj řídící jednotky vlastní konstrukce je nedostatečná funkcionalita stávající řídící jednotky a obslužného softwaru dodávaného výrobcem lineárního motoru. Zásadním nedostatkem bylo omezení počtu instrukcí prováděného programu. Dle sdělení zadavatele neumožňuje dodávaná aplikace provádět složitější pohyby z důvodu omezení počtu prováděných instrukcí.

Původní řídící jednotka ESMC–C2 [2] má na výběr mezi režimem Controler (kdy jsme limitováni možnostmi obslužného softwaru) a režimem Driver, kdy prostřednictvím elektrických impulsů tvořených podle dokumentace můžeme prostřednictvím původní řídící jednotky přímo ovlivňovat chod motoru (posun, ovládání magnetické brzdy a bezpečnostních prvků, čtení stavových informací).

Při použití režimu Driver jsme tedy limitováni pouze vlastními schopnostmi při vývoji vlastní řídící jednotky a obslužného softwaru. Vlastnosti a funkcionalita řídící jednotky jsou popsány v externím dokumentu. Tento dokument však neobsahuje naprosto kompletní seznam všech vlastností, protože některé kroky v analýze a realizaci vyžadovaly přidání zcela nových požadavků nebo omezení, případně docházelo k jejich drobným úpravám.

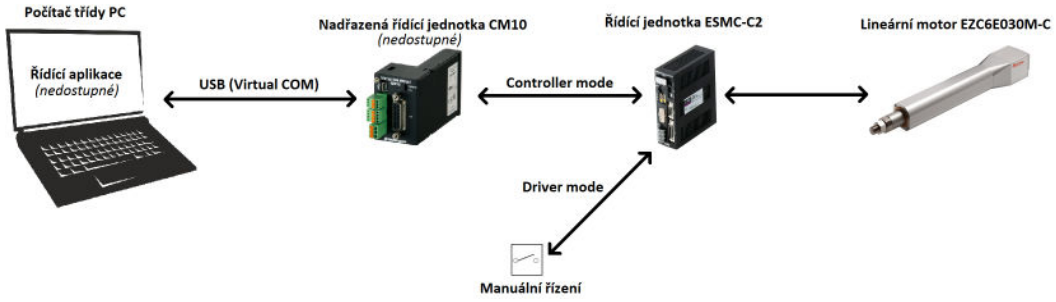
Cílem této práce je vytvoření nové řídící jednotky a obslužného softwaru podle písemného a slovního technického zadání dodané Laboratoří biomechaniky člověka a katalogových listů, manuálů a jiné dokumentace dodávané výrobcem lineárního motoru a původní řídící jednotky, případně dokumentace k dalším použitým zařízením, které jsou nezbytné pro správnou činnost.

Celý systém by měl implementovat co největší počet funkcí a měl by dosahovat vysoké spolehlivosti a přesnosti, aby naměřené výsledky byly co nejméně zatížené chybami, popřípadě aby se zamezilo nechtěnému zničení měřených biologických a syntetických vzorků. Z tohoto důvodu by měl být celý systém predikovatelný, neboť problémy se budou muset řešit automaticky a pokud možno v reálném čase.

Systém by měl být řízen a spravován z výpočetního prostředí MATLAB. Dále by měl spolupracovat s některými dalšími přístroji v laboratoři, jako jsou například vysokorychlostní kamery a různé druhy senzorů pro měření fyzikálních veličin.

2.1 Stav při zahájení práce

Na obrázku 2.1 je zřejmý stav celého systému před zahájením práce. Obdržel jsem lineární motor EZ Limo EZC6E030M–C a jeho řídící jednotku ESMC–C2. K řídící jednotce ESMC–C2 jsem dostal tištěnou dokumentaci [2]. Bohužel jsem neměl možnost se blíže seznámit s nadřazenou řídící jednotkou CM10 ani s obslužnou řídící aplikací dodávanou výrobcem motorů a řídících jednotek. Při zakreslení nedostupných částí jsem vycházel ze slovních informací pracovníků laboratoře a částečně z problematicky dostupných manuálů a katalogových listů výrobce.



Obrázek 2.1: Stav při zahájení práce

2.2 Soupis požadované funkčnosti – Technického zadání

V této kapitole rozeberu jednotlivé požadavky na softwarové a hardwarové části. Podle požadavků v zadávácí dokumentaci je nutné implementovat velké množství příkazů v systému MATLAB. Tyto funkce jsou ve formátu vektor s návratovými hodnotami, příkaz, vstupní parametry.

`(vektor s návratovými hodnotami) = PŘÍKAZ(vektor vstupních parametrů)`

Níže uvádíme úplný seznam všech požadovaných funkcí, které mají být implementovány v systému MATLAB.

2.2.1 Příkazy pro realizaci pohybu

Představa zadavatele je využít knihovní funkce pro ovládání nejrůznějších motorů používaných v laboratoři. Cílem bylo sjednotit ovládání všech potenciálních motorů.

(xr,tr,v,error) = MOVE(unit,x,t)

Tato funkce provede posun aktuátoru na zadanou pozici v zadaném čase.

Vstupní parametry:

1. unit = 1,2,3 (výběr aktuátoru)

2. x – požadovaná poloha [mm] (v absolutním s.s., kde výchozí poloha je definována funkcí SETX)
3. t – požadovaný čas [s] (absolutní čas, vzhledem k počátku definovaném funkcí SETIME)

Výstupní parametry

1. xr – skutečná poloha [mm] (v absolutním s.s., hodnota je dopočítávána z celkového počtu požadovaných kroků krokového motoru od výchozí polohy definované funkcí SETX)
2. tr – skutečný čas, kdy byl ukončen posuv [s]
3. xrv – vektor skutečné polohy [mm]
4. trv – vektor času, kdy byl ukončen posuv [s]
5. v – průměrná rychlost [mm/s]
6. error
   - 0 – vše je v pořádku
   - 1 – nerealizovatelná podmínka pro čas, příliš vysoká požadovaná rychlost
   - 2 – finální požadovaný čas menší než aktuální zjištěný z hodin počítače
   - 3 – koncový spínač 1
   - 4 – koncový spínač 2
   - -1 – jiná chyba, např. nezapnutý kontrolér krokových motorů

Tato generická varianta příkazu ještě může mít tyto varianty:

(xr,tr,v,error) = MOVEV(unit,x,v)

Stejná jako předchozí varianta, jen se zadává rychlost a ne koncový čas.

(xr,tr,v,error) = MOVEA(unit,x,v)

Stejná jako předchozí, ale s inteligentním nastavením rychlosti (minimalizace zrychlení, případ, kdy zadáváme extrémně vysoké rychlosti a dlouhý časový krok).

(xr,tr,v,error) = MOVEVEKTOR(unit,x,t)

Stejná jako MOVE jen se zadává vektor polohy a vektor času.

2.2.2 Příkazy pro konfiguraci a čtení stavových informací

(xr,error) = READ(unit)

Tato funkce neaktivuje krokové motory, jen předá vypočtenou absolutní hodnotu x a vyhodnotí signály koncových spínačů.

error = SETIME(unit,tini)

tini – počáteční absolutní čas [s] (přečte se aktuální čas hodin počítače a tomu se přiřadí tini – obvykle nula).

error = SETX(unit,xini)

xini – počáteční poloha [mm] (počátek absolutního souřadného systému – obvykle nula).

(vystup) = HELP(error)

Funkce HELP vyhodnotí error a na obrazovce zobrazí informaci o selhání.

- 1 – nerealizovatelná podmínka pro čas, příliš vysoká požadovaná rychlost
- 2 – finální požadovaný čas menší než aktuální zjištěný z hodin počítače
- 3 – koncový spínač 1
- 4 – koncový spínač 2
- -1 – jiná chyba, např. nezapnutý kontrolér krokových motorů

2.2.3 Variace příkazů MOVE pro speciální činnost

Koncové spínače (funkci lze volat až po aktivaci funkce SETX). Alternativně identifikace poloh obou snímačů současně (pohodlnější pro uživatele).

(xr,error) = SEND(unit,v)

v – rychlost [mm/s] (kladná hodnota posun směrem rostoucího x až do vypnutí koncovým spínačem 1, záporná hodnota posun zpátky až ke sepnutí spínače 2).

(xmin,xmax,error) = SEND2(unit,vabsutní)

Alternativně identifikace poloh obou snímačů s odhadnutou polohou koncových spínačů a opatrným dojezdem (přesnější zjištění polohy)

(xmin,xmax,error) = SEND2A(unit,xmine,xmaxe,vfast,vslow)

Vstupní parametry xmine, xmaxe, kde se provádí rychlý pojezd rychlostí vfast, musí být subintervalem xmin, xmax.

2.2.4 Příkazy, které nejsou v seznamu

Tyto příkazy byly přidány během postupného zpřesňování zadání, popřípadně byly jako požadavek na obecnou funkčnost. Takovým příkladem je požadavek, aby lineární motor bylo možné kdykoliv bezpečně zastavit uživatelem (stisk klávesy na klávesnici), při jakékoliv nestandardní reakci systému (failsafe) a aby tento systém byl reaktivní na podněty od nejrůznějších senzorů a ovládacích prvků systému.

2.2.4.1 Synchronizační příkaz pro spuštění řídící jednotky

Pro spolupráci s ostatními přístroji v laboratoři je nutné výsledný systém vybavit externím synchronizačním portem (trigger), kterým se při výskytu napětí 5 V uvedou všechna zařízení do pracovních režimů (měření bude spuštěno současně na všech přístrojích).

2.2.4.2 Synchronizační příkaz pro zastavení řídící jednotky

Pro obsluhu asynchronních událostí od senzorů (koncové dorazy, tenzometry, tlakoměry, teplotní čidla) je nutné implementovat systém externích portů (portu), který v případě požadované události bezpečně vypne motor. Tento požadavek by měl zabránit škodám při výskytu nestandardních pracovních podmínek (například zvýšená teplota okolí může ovlivnit měření nepříznivým způsobem).

Tento mechanizmus by měl zároveň sloužit i k uživatelskému vypnutí motoru. Jde tedy o obdobu „generálního stopu“.

2.2.4.3 Příkazy pro ovládání magnetické brzdy

Lineární motor EZ Limo EZC6E030M–C je vybaven magnetickou brzdou, která umožňuje při aktivaci odpojení buzení motoru a umožní manuální pohyb pohyblivé části lineárního motoru. Lze tedy před začátkem měření ručně nastavit výchozí polohu.

Je tedy nutné implementovat i příkazy, které budou realizovat sepnutí a vypnutí magnetické brzdy.

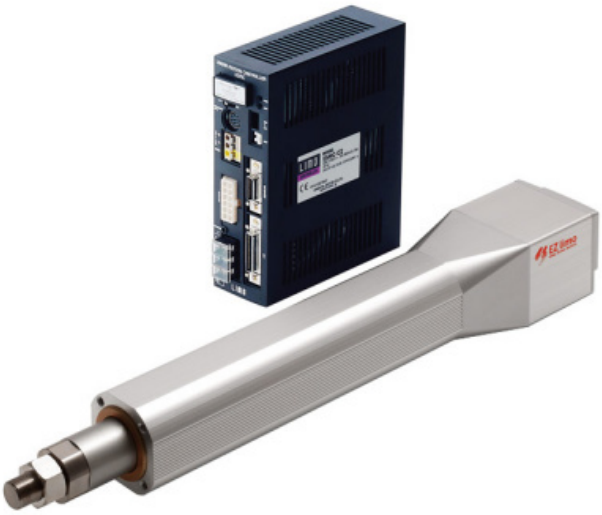
2.3 Lineární motor EZ Limo EZC6E030M–C

Poslední částí analýzy celého systému je rozbor funkcionality lineárního motoru EZ Limo EZC6E030M–C (na obrázku 2.2). Každý motor tohoto výrobce je dodáván s kompatibilní řídící jednotkou (v našem případě ESMC–C2), pro kterou je dodáván hardwarový klíč obsahující klíčové parametry (tabulka 2.1) připojovaného motoru.

Při zkoumání dokumentace k dalším motorům a řídícím jednotkám toho výrobce jsem zjistil, že velká část portfoilia výrobce používá stejné komunikační protokoly, a proto jsem si dal za cíl navrhnout a realizovat řídící systém více obecně, aby tento systém mohl být využit pro více různých motorů a řídících jednotek.

Parametry lineárního motoru EZ Limo EZC6E030M–C	
Délka pohyblivé části	300 mm
Rozlišení	0.01 mm
Přesnost při opakovaném návratu na stejnou pozici	±0.02 mm
Maximální rychlost	300 mm/s
Maximální síla tahu	400 N
Maximální síla tlačení	500 N
Magnetická brzda	Ano

Tabulka 2.1: Pararametry lineárního motoru EZ Limo EZC6E030M–C



Obrázek 2.2: Lineární motor EZ Limo EZC6E030M–C s řídící jednotkou ESMC–C2

2.4 Řídící jednotka ESMC–C2

Dodávanou řídící jednotkou k lineárnímu motoru EZ Limo EZC6E030M–C je driver ESMC–C2. Řídící jednotka se stará o řízení napájecích obvodů lineárního motoru, kdy driver funguje jako spínaný zdroj o přibližném příkonu 700 W. Další částí driveru je výpočetní část, která se stará o překlad řídících příkazů. Tato řídící jednotka podporuje 2 režimy obsluhy. Manuál k řídící jednotce ESMC–C2 je k dispozici na webu výrobce [2].

2.4.1 Podporované režimy řídící jednotky

- Controller mode — Pro tento mód je určena nadřazená řídící jednotka CM10. Tu jsem v době práce na diplomové práci neměl k dispozici, a proto jsem vycházel jen ze znalostí kolegů z Laboratoře biomechaniky člověka Fakulty strojní.

Řídící jednotka ESMC–C2 tak pro jednotku vykonává zasílané instrukce. Většina vstupních a výstupních signálů je tak určena pro různá monitorovací měření (opuštění zóny s dovoleným pohybem, koncové dorazy, alarm).

- Driver Mode — Režim pro přímé řízení řídící jednotky. K ovládání všech funkcí je potřeba generovat digitální signály s požadovaným tvarem a časováním. Přehled nejdůležitějších signálových vstupů a výstupů (tabulky 2.2 a 2.3):

Vstupní signály	
ACL/CK	Smazání příznaku alarmu. Alternativní funkce je hodinový kmitočet pro vyčítání stavových informací z řídící jednotky.
FREE	Uvolnění magnetické brzdy a odpojení napájecího napětí od motoru shodné se vstupem C.OFF.
C.OFF	Odpojení napájecího napětí od motoru.
HMSTOP	Zastavení operace návratu na počáteční pozici.
HOME/PRESET	Spuštění operace návratu na počáteční pozici. Alternativní funkce je výstup aktuální polohy při žádosti o ni.
REQ	Žádost o stavové informace.
FP+/FP-	Vykonání pohybu o jeden krok vpřed nebo vzad.

Tabulka 2.2: Vstupní signály řídící jednotky ESMC–C2

Výstupní signály	
ALM	Příznak alarmu při nestandardní nebo nedovolené operaci.
MOVE	Signál indikující pohyb motoru.
END/OUTR	Komplementární signál k příkazu MOVE. Dále indikuje dokončení příkazu HOME.
OUT0	Určeno ke přečtení polohy při žádosti o ni. První výstup.
OUT1	Určeno ke přečtení polohy při žádosti o ni. Druhý výstup.
ASG1/ASG2	Výstupy kvadraturního enkodéru. Jde o nezávislý elektromechanický senzor pohybu motoru.

Tabulka 2.3: Výstupní signály řídící jednotky ESMC–C2

2.4.2 Elektronické zapojení řídící jednotky

Řídící jednotka ESMC–C2 má galvanicky oddělený systém vstupů a výstupů, které jsou realizovány opto–elektrickým způsobem. Typický vstup do motoru je řešen pomocí proudové smyčky procházející přes optočlen (LED dioda – fototranzistor), která se sepne jiným optočlenem do země.

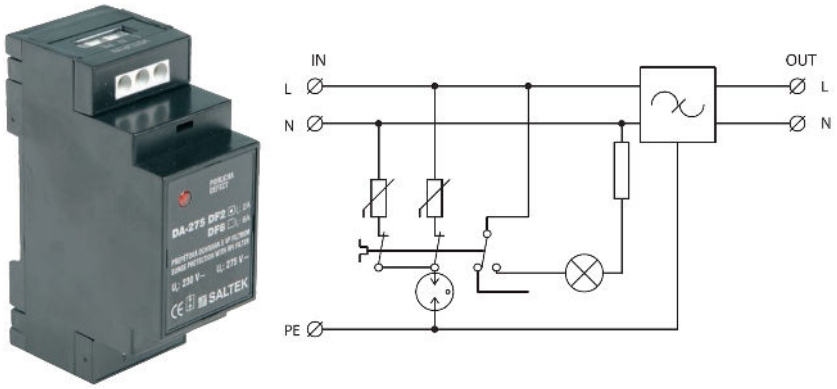
Podobný princip je realizován pro výstupy řídí jednotky, kdy řídící jednotka ESMC–C2 uzavírá proudovou smyčku přes optočleny na straně nové řídící jednotky. Z pohledu funkce jde o zapojení vstupu s otevřeným kolektorem.

Z obrázku elektronického zapojení (v příloze A.1) je dobře patrné galvanické oddělení obou řídících jednotek. Tato část řídící jednotky ESMC–C2 tedy vyžaduje separátní napájecí zdroj pro napětí 24 V. Celá soustava řídící jednotky ESMC–C2 a lineárního motoru EZ Limo EZC6E030M–C vyžaduje ke své funkčnosti tato napájecí napětí:

- Napájení motoru — 230 V AC @ 3 A
- Napájení logické části řídící jednotky — 24 V DC @ 1 A
- Napájení galvanicky oddělené části — 24 V DC @ 0.2 A

Kromě toho bude muset mít vlastní zdroj i nová řídící jednotka z důvodu galvanického oddělení systémů. V celkové konstrukci celého systému aplikace bude nutné pořídit nejméně tři různé napájecí zdroje.

Dále v požadavcích pro zajištění elektromagnetické kompatibility [2] je velmi doporučeno použít přepěťové ochrany a síťového vysokofrekvenčního filtru. Všechny kabely, včetně napájecích, by podle doporučení výrobce měly být stíněné. Při výběru jsem se nakonec nerozhodl použít výrobcem doporučené přepěťové ochrany a VF síťový filtr jako jednotlivá zařízení a rozhodl jsem se použít výrobek renomované české firmy Saltek s.r.o., která vyrábí zařízení kombinující obě tyto funkce. Konkrétně jde o síťový VF filtr s integrovanou přepěťovou ochranou III. třídy DA-275 DF [12] s montáží na standardní DIN lištu (na obrázku 2.3).



Obrázek 2.3: Vzhled a vnitřní zapojení kombinované přepěťové ochrany a síťového VF filtru Saltek DA-275 DF [12]

Kromě těchto částí je nutné dodržet náběhové doby napájecích napětí pro jednotlivé části systému a pro dodržení požadavků z hlediska bezpečnosti práce přidat i další bezpečností prvky, například jističe nebo proudové chrániče.

Kapitola 3

Analýza a návrh řešení

Prvním krokem bylo prostudování dostupné dokumentace k lineárnímu motoru a řídící jednotce. Z dokumentace vyplynulo, že bude nezbytně nutné seznámit se s těmito technickými obory:

- Informatika – reverzní inženýrství, návrh řídícího systému a jejich implementace
- Numerická matematika – MATLAB a transformace spojitých trajektorií na diskrétní
- Mechanika – činnost lineárního motoru, samotná fyzická montáž, testování a ověřování korektnosti implementovaných funkcí
- Elektrotechnika – elektrická zapojení, tvorba desky plošných spojů, výběr vhodných součástek, zdrojů, přístrojů, součinnost s ostatním vybavením laboratoře, odolnost vůči rušení a EMC (elektromagnetická kompatibilita)

Po prostudování i technického zadání, zejména požadavků na funkcionalitu jsem se mohl rozhodnout pro tři možné přístupy:

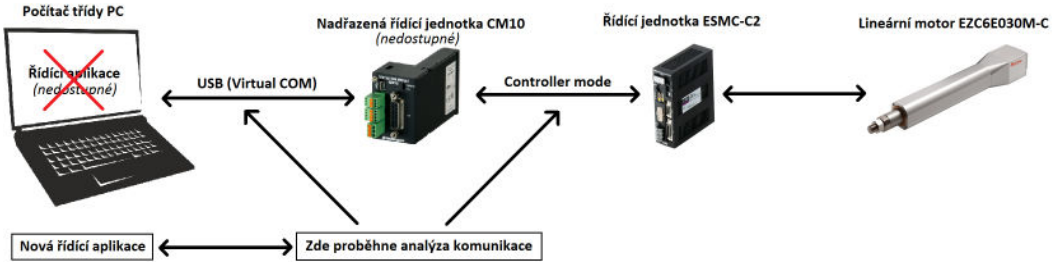
- Pokusit se o reverzní inženýrství režimu Controler a obejít tak omezení na velikost programu pro původní řídící jednotku. Výsledkem by měla být nová řídící aplikace.
- Navrhnout celý systém řídící jednotky od začátku včetně výkonových částí pro buzení lineárního motoru.
- Využít z původního systému pro řízení lineárního motoru co nejvíce částí z důvodu poměrně jednoduchého rozhraní původní řídící jednotky (režim Driver) a nenavrhovat si vlastní výkonovou část řídící jednotky.

3.1 Přístup k tvorbě nového řídícího systému

3.1.1 Reverzní inženýrství

Myšlenka na reverzní inženýrství vycházela z poznatku, že zásadní nedostatek původního řídícího systému je omezení délky vykonávaného programu. Původní aplikace má zásadní

nedostatek v délce prováděného programu a nespolupracuje se systémem MATLAB. Vývojem nové aplikace bylo by možné tato omezení obejít. To by zjednodušilo vývoj, neboť technické zadání je založeno na funkcionalitě původního softwaru dodávaného výrobcem. Předpokládaný stav při použití reverzního inženýrství je na obrázku 3.1.



Obrázek 3.1: Předpokládaný stav při použití reverzního inženýrství

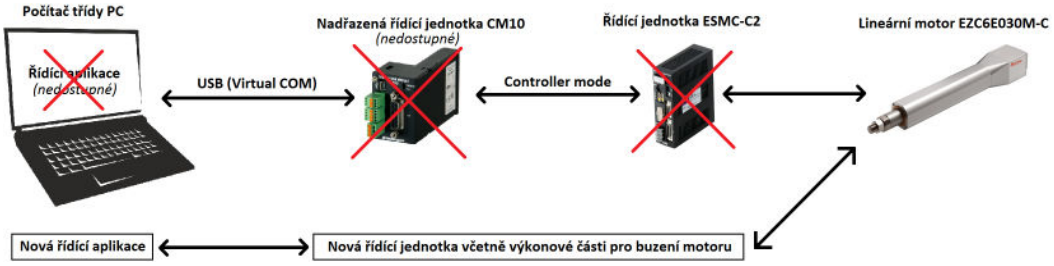
Problémem se však ukázala neexistence dokumentace ke komunikaci mezi řídící jednotkou a obslužným softwarem, stejně tak i fyzická nedostupnost výše zmíněných částí. Komunikace s řídící jednotkou ESMC–C2 je realizována prostřednictvím 32–bitového paralelního rozhraní. Některé signály sice pravděpodobně souhlasí s dokumentací, přesto sledování a porozumění komunikace na tomto množství vodičů není jednoduché. Vysledování celého instrukčního souboru a jejich závislostí v rozumné době tak není prakticky možné.

Vzhledem k nutnosti použití nadřazené řídící jednotky CM10 (obrázek 3.2) pro režim Controller řídící jednotky ESMC–C2 by přibyla nutnost analyzovat i komunikační protokol jednotky CM10. Tuto řídící jednotku jsem však neměl k dispozici, a proto jsem myšlenku na reverzní inženýrství zavrhl.

Obrázek 3.2: Nadřazená řídící jednotka CM10 pro řídící jednotku ESMC–C2

3.1.2 Návrh vlastního řídícího systému od začátku

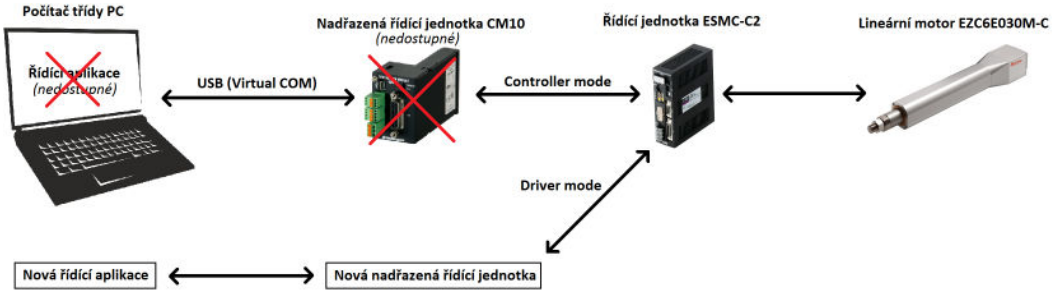
Druhou možností je návrh celé nové řídící jednotky od začátku. Protože se však neorientuji ve spínaných zdrojích a výkonových prvcích obecně, tuto možnost realizace jsem zavrhl. Stejně tak mi není známo elektrické zapojení motoru, bez jehož znalosti nemohu motor připojit k výkonovým částem nové řídící jednotky. Předpokládaný stav při návrhu vlastního řídícího systému od začátku je znázorněn na obrázku 3.3.



Obrázek 3.3: Předpokládaný stav při návrhu vlastního řídícího systému od začátku

3.1.3 Návrh vlastního řídícího systému s využitím stávajících částí

Třetí možností je návrh nové a komplexní řídící jednotky, která by využívala režim Driver původní řídící jednotky a tím bylo dosaženo maximální možné kontroly nad jednotlivými funkcemi systému. Tím obejdeme původně hlavní nedostatek původní řídící jednotky spočívající v omezené délce programu pro lineární motor. Předpokládaný stav při návrhu vlastního řídícího systému s využitím stávajících částí je znázorněn na obrázku 3.4.



Obrázek 3.4: Předpokládaný stav při návrhu vlastního řídícího systému s využitím stávajících částí

Hlavní nevýhodou tohoto přístupu je vytvoření nadřazené řídící jednotky od začátku. Výhodou je znalost vnitřního fungování a z toho plynoucí možnost predikovatelnosti chování navrženého systému. Další výhodou je možnost implementace v prostředí (technologii), se kterým máme zkušenosti a urychlit tak návrh a tvorbu řídícího systému.

3.1.4 Volba přístupu k tvorbě nadřazené řídící jednotky

Z konstatovaných skutečností jsem se rozhodl pro návrh vlastní řídící jednotky s využitím již existujících částí, kterou jsem popsal v předchozí kapitole 3.1.3.

3.2 Analýza požadované funkčnosti a možné přístupy realizace

Ze seznamu požadavků je patrné, že půjde o poměrně komplexní systém, který má zabezpečit správné provádění všech příkazů a funkcí, být odolný vůči chybám, reaktivní na podměty uživatele i okolního prostředí. Dále má spolupracovat s ostatními přístroji v laboratoři. Činnost systému by měla být predikovatelná.

V zadání se vyskytují dva hlavní typy příkazů, příkazy pohybu (posunu) a příkazy pro konfiguraci a čtení stavových slov.

3.2.1 Architektura systému

Celý systém je možno realizovat dvěma základními přístupy:

1. **Monolitická architektura**

   Monolitická architektura systému realizuje veškeré funkce v jedné úrovni. Tato architektura se objevuje u systémů, které bez mezikroků vykonávají požadované funkce.

   V našem konkrétním příkladu by to znamenalo, že systém by rovnou interpretoval příkazy systému MATLAB. Takový systém by však byl neúměrně složitý, neboť by musel implementovat funkčnost MATLABu i samotné řízení hardwaru pro každý jednotlivý příkaz.

2. **Hierarchická architektura**

   Tato architektura se využívá u každého složitějšího problému. Je přímým důsledkem metody Divide & Conquer (rozděl a panuj), kdy se komplexní problém dělí na podproblémy, dokud nejsme schopni elementární podproblémy vyřešit. Z řešení dílčích podproblémů se pak sestaví celé řešení.

   To odpovídá rozdělení funkčnosti systému na několik vrstev, kdy každá řeší specifickou činnost a využívá služeb nižších vrstev. Příkladem může být síťový model ISO/OSI, který obsahuje 7 různých vrstev, mezi kterými probíhá komunikace.

Jak jsem zmínil v úvodu kapitoly 3.2, jde o realizaci komplexního systému, je vhodné použít hierarchickou architekturu, kde rozdělím činnost celého systému do komunikujících vrstev. Každá z těchto vrstev může být realizována na hardwarové nebo softwarové úrovni. V následujícím textu se pokusím o dekompozici systému metodou „shora — dolů“.

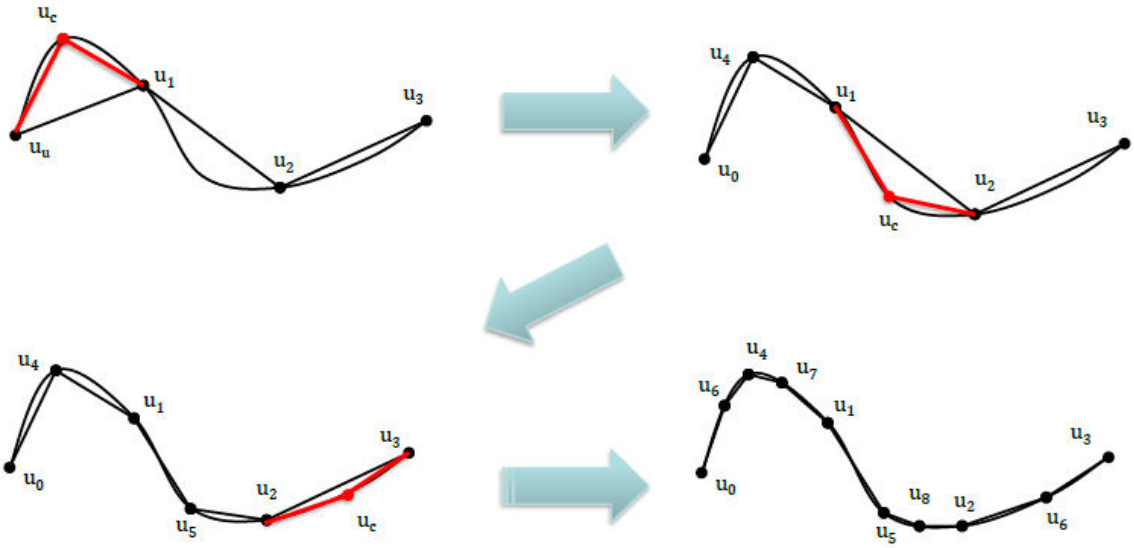
3.2.2 Dekompozice systému na jednotlivé vrstvy metodou „shora — dolů“

Řešeným (a dekomponovaným) problémem je převod volání MATLABovských funkcí na pohyb vykonávaný lineárním motorem. Takový problém lze vyřešit pouze velmi komplikovaně a je nutná dekompozice.

Samotná dekompozice je proces, kdy jednotlivé části (úrovně) systému mohou vznikat a zanikat během procesu analýzy a realizace. Proto jsem rozhodl zde popsat až téměř kompletně dekomponovaný systém.

3.2.2.1 Úroveň „vysoké“ numerické matematiky

V této vrstvě bude probíhat překlad a práce s MATLABovskými příkazy, které jsou zadány v technickém zadání. Na této úrovni bude řešen převod obecných křivek (realizace zadaných trajektorií) na křivku [16], která bude diskrétní v čase a velikosti (znázorněno na obrázku 3.5). Nutnost diskretizace vychází z vlastností lineárního motoru a jeho řídící jednotky, neboť maximální možná rychlost posunu je 300 mm/s a minimální možný krok motoru je 0,01 mm. Z toho vyplývá rychlost 30 kHz a tu lze převést na čas 0,33 μs.



Obrázek 3.5: Příklad diskretizace křivky na soustavu vektorů, převzato z [16]

Z příkladu diskretizace křivky na vektory je patrné, že množina příkazů pro pohyb motorů bude využívat jako nižší vrstvu variantu příkazu MOVE, kdy se zadává vektor udávající rychlost a vzdálenost výsledného pohybu.

3.2.2.2 Úroveň „nízké“ numerické matematiky

Na této úrovni je nutné řešit převod datových struktur MATLABu na datové struktury používané v hardwaru. Je zde možnost pracovat a posílat informace v plovoucí desetinné čárce, popřípadě posílat data v celočíselném formátu.

Výhodou zpracování hodnot reprezentovaných v plovoucí desetinné čárce je jednoduchost implementace z pohledu MATLABu, nevýhodou je pomalé zpracování v hardwaru a jeho nepřesnost, která způsobena reprezentací desetinných čísel v binární soustavě.

Při použití celočíselných datových typů je potřeba vymyslet vhodný převod desetinných čísel do vhodného formátu, aniž by tím utrpěla funkčnost celého zařízení. Celočíselné datové typy však mají velkou výhodu v podobě snadného a rychlého zpracování v hardwaru.

Protože ovládání je založeno na PWM (Pulse Width Modulation), je možné čas operace převést na frekvenci. Samotná frekvenční dělička se dá snadno realizovat čítačem o dostatečné velikosti při velmi malé odchylce od času vyjádřeného ve formátu využívají desetinnou čárku.

S ohledem na snadnou realizaci jsem zvolil 32–bitové celočíselný formát pro komunikaci a zpracování, neboť 32–bitové celé číslo odpovídá na standardní PC architektuře datovému formátu Integer.

3.2.2.3 Úroveň řízení systému

V této části se bude realizovat předávání vypočítaných informací do úrovně systému (software/hardware), který bude tyto informace interpretovat na signály, kterými se bude ovládat původní řídící jednotka lineárního motoru.

Dalším funkčním požadavkem je obsluha asynchronních událostí, které bude generovat jak uživatel, tak nižší vrstvy systému.

V této vrstvě bude pravděpodobně realizován přechod mezi softwarovou a hardwarovou částí řídícího systému.

3.2.2.4 Úroveň tvorby signálů

Na této úrovni se budou přijaté informace (vektory polohy a času) transformovat na elektrické signály pro další použití a reagovat na vstupy od původní řídící jednotky. Bude zde realizováno řízení na nejnižší úrovni.

3.2.2.5 Úroveň komunikace s původní řídící jednotkou

Nejnižší vrstva systému slouží k přizpůsobení elektrických parametrů celého systému a původní řídící jednotky. Jde především o přizpůsobení napěťových úrovní, galvanické oddělení systému a zajištění elektromagnetické kompatibility (EMC) všech dílčích částí a určitou úroveň mechanické odolnosti splňující alespoň základní požadavky na bezpečnost zařízení.

3.2.3 Funkčnost realizovatelná softwarovou cestou

Při realizaci systému jsem vybral tyto vrstvy pro realizaci v softwaru:

- Úroveň „vysoké“ numerické matematiky
- Úroveň „nízké“ numerické matematiky
- Úroveň řízení systému

Pro tyto vrstvy je společné, že budou s výhodou těžit z vysokého výkonu počítače, který bude numerickými výpočty provádět diskretizaci obecné křivky pohybu na vektory času a vzdálenosti. Stejně tak bude pro obsluhu zařízení jednoduší pracovat a připravovat programy pro motor na standardním počítači typu PC.

Prvotní nástřel systému a testování použitých algoritmů bude z důvodu jednoduchosti a spolehlivosti implementováno v programovacím jazyce C/C++. Po validaci všech funkcí systému budou tyto algoritmy přepsány do prostředí MATLAB. Z tohoto důvodu musí být rozhraní a protokol pro komunikaci podporovaný a snadno realizovatelný v prostředí MATLAB, C/C++ a v samotné hardwarové části systému.

3.2.4 Funkčnost realizovatelná hardwarovou cestou

Při realizaci systému jsem vybral tyto vrstvy pro realizaci v hardwaru:

- Úroveň řízení systému
- Úroveň tvorby signálů
- Úroveň komunikace s původní řídící jednotkou

Tyto úrovně systému mají společnou vlastnost, že je na standardním počítači typu PC nelze bezproblémově realizovat, případně nejdou realizovat vůbec. Úroveň řízení se vyskytuje v softwarové i hardwarové části, z čehož je patrné, že bude fungovat jako spojovací můstek mezi softwarovou a hardwarovou částí řídícího systému. Z toho vyplývá, že obě tyto části musí spolu komunikovat některým standardizovaným způsobem.

Pro nejnižší 2 vrstvy systému bude nutné navrhnout desku plošných spojů (DSP), která bude obsahovat nějakou formu mikrořadiče, realizované pomocí jednočipového mikroprocesoru nebo FPGA/CPLD čipu. Spolu s tím bude deska plošných spojů realizovat přizpůsobení napěťových úrovní pro komunikaci s původní řídící jednotkou lineárního motoru.

3.3 Výběr platformy pro realizaci

Při výběru platformy pro realizaci postupuji naopak směrem „zdola — nahoru“, protože v případě neimplementování některé funkcionality je nutné zajistit její implementaci ve vrstvách nadřazených. Snahou je však maximálně efektivní využití všech vlastností a výhod, které poskytuje konkrétní vrstva systému.

Jako první krok je nutné udělat analýzu možných platforem pro realizaci hardwarové části řídícího systému.

3.3.1 Důležitá a časově kritická funkcionalita

Režim Driver obsahuje celkem 7 různých výstupních řídících signálů, které lze považovat za asynchronní. Za nejdůležitější je možné považovat výstupy z kvadraturního enkodéru. Tyto výstupy nemají pevně stanovený kmitočet, který se může pohybovat mezi 0 Hz až po přibližně 150 kHz. Z této rychlosti vyplývá nutnost použít rychlost vzorkování na více než dvojnásobné

rychlosti výstupů podle vzorkovacího (Shannonova) teorému. Protože řídící jednotka má být univerzální pro více typů motorů, odhadl jsem spodní mez vzorkovacího kmitočtu na 5 MHz. Výstupy z kvadraturního enkodéru je tedy nutné zpracovat čistě na úrovni hardwaru. V průmyslovém prostředí se běžně používají pro dekódování signálů programovatelné logické obvody typu CPLD.

3.3.2 8–mi a 16–ti bitové jednočipové mikropočítače

První možností je realizace systému na jednoduchých jednočipových mikropočítačích (typické mikropočítače od firem Atmel AVR/MicroChip PIC). Jejich výhodou je nízká cena, dobrá dokumentace a jednoduchost vývoje. Nevýhodou je malý počet vstupně–výstupních pinů (I/O pins), nízký pracovní kmitočet mikropočítačů (jednotky MHz), pouze sekvenční zpracování a ošetření případných chybových stavů.

Z těchto vlastností je patrné, že takto realizovaný systém je snadné zahltit vnějšími podměty, a proto ho nelze použít jako spolehlivý reaktivní systém. Spolu s tím mají tyto mikropočítače jen velmi málo paměti, a proto není možné implementovat frontu příkazů, která by zajišťovala plynulé zpracování příkazů a tím i plynulý pohyb motoru.

3.3.3 32–ti bitové jednočipové mikropočítače třídy ARM

Další možností je využít jednočipového mikropočítače, který obsahuje procesorové jádro třídy ARM. V současné době se využívá několik různých typů těchto jader (ARM7, ARM9, Cortex–Ax, Cortex–Mx). Pro naše použití je považujeme za ekvivalentní, neboť od nich nebudeme vyžadovat žádnou speciální funkcionalitu.

Tyto obvody mají velmi výkonné procesorové jádro, některé z nich jsou vybaveny i dedikovanou jednotkou pro práci s čísly v plovoucí desetinné čárce. Všechny tyto mikropočítače jsou vybaveny velkým množstvím periferních obvodů, kterými lze komunikovat s okolním světem. Velmi častá je i realizace USB Endpointu jako periferního obvodu mikropočítače. Tím by bylo možné realizovat propojení k PC prostřednictvím USB (Univerzální Sériová Sběrnice).

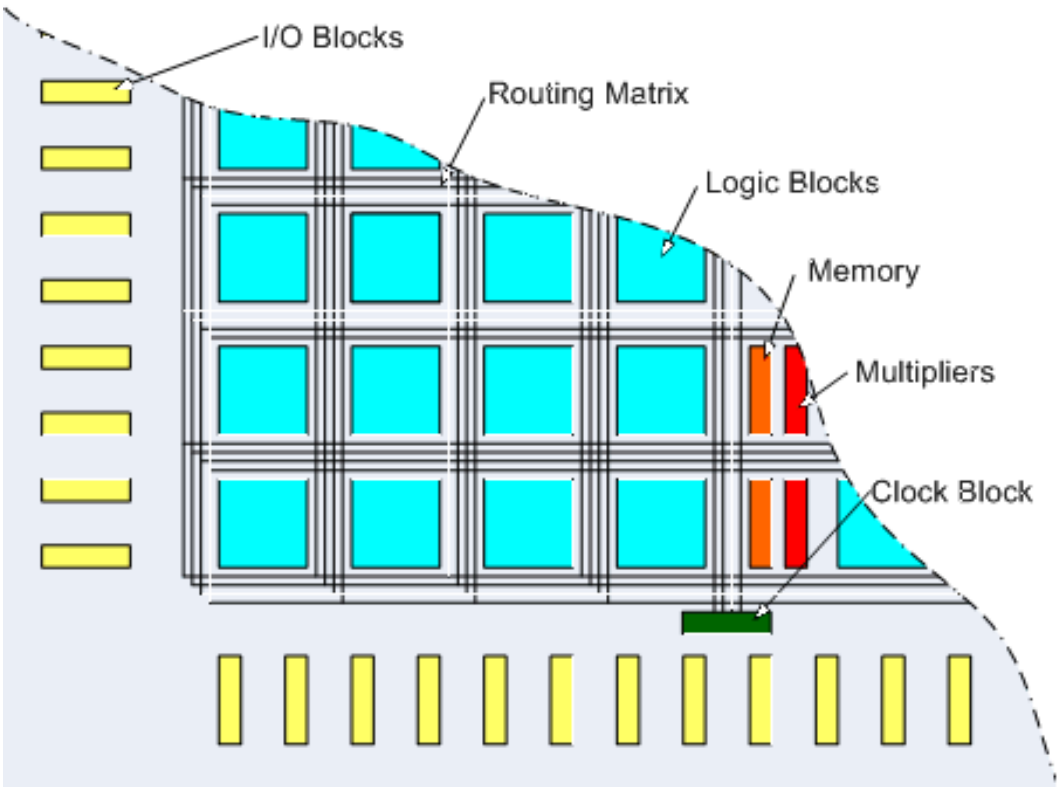
Jejich nevýhodou je horší dokumentace, z níž vyplývá nemožnost přesně predikovat chování mikropočítače vlivem vyrovnávacích pamětí a vlivem překladu programu z jazyka C do strojového kódu, kdy překladač může provádět nejrůznější optimalizace. I přes vysokou pracovní frekvenci jádra (stovky MHz) jsou periferní obvody taktované mnohem pomaleji (jednotky MHz) za účelem snížení spotřeby celého mikropočítače. To však zhoršuje využití vstupně/výstupních pinů za účelem generování signálových průběhů pro původní řídící jednotku. Dále pro plné využití mikropočítače jsou nutná poměrně drahá vývojová prostředí.

3.3.4 Programovatelná hradlová pole třídy FPGA

Poslední z uvažovaných možností je nasazení obvodů programovatelných hradlových polí (FPGA — Field Programable Gate Array). Tyto obvody nejsou klasickým integrovaným obvodem s jednou charakteristickou funkcí (procesor, paměť, A/D a D/A převodníky), ale obsahující velký počet bloků, kterými lze reprezentovat výrazy v matematické logice a provádět nad nimi operace v Booleově algebře. Tyto bloky jsou spojeny propojovací sítí, která

dovoluje vytváření složitějších logických funkcí, než by bylo možné na jednom elementárním bloku.

Kromě těchto dvou základních částí mohou FPGA čipy obsahovat bloky realizující paměť RAM, bloky DSP (Digitální signálový procesor) pro rychlé zpracování číslicových signálů. Dále může obsahovat již hotová jádra procesorů (Sparc, ARM) a násobičky a děličky hodinových signálů (DCM a PLL) [4]. Počet všech těchto bloků realizovaných na FPGA je přímo úměrná použité výrobní technologii a ceně použitého čipu FPGA.



Obrázek 3.6: Struktura FPGA čipu, převzato z [4]

Výhodou FPGA je možná plná paralelizace prováděných algoritmů. Protože jsou algoritmy psány na nejnižší možné úrovni, máme plnou kontrolu nad výslednou strukturou uvnitř FPGA a proto můžeme v rámci možností garantovat odezvy systému ve stanovených časových omezeních.

Tyto algoritmy jsou poměrně jednoduché a nelze kontroly těchto algoritmů žádným způsobem obejít, což je vhodné pro ošetření nejrůznějších chybových stavů. Nevýhodou může být prvotní pochopení paralelního programování a nutnost vytvořit řídící systém na míru z elementárních struktur.

3.3.5 Vybraná platforma pro realizaci

Ze tří výše popsaných variant jsem se rozhodl realizovat řídící systém na FPGA, protože jednoduché jednočipové mikropočítače i složitější mikropočítače třídy ARM měly pro můj záměr více negativních vlastností než čipy FPGA. FPGA je nutné pro možnost přesného časování všech součástí systému s možností garantovat odezvu s přesností na jeden hodinový takt. Při použití čipu FPGA není nutné používat pro dekódování signálu z kvadraturního enkodéru přídavný obvod CPLD, neboť jeho funkčnost můžeme realizovat v FPGA.

Svoji roli hrály i moje dosavadní zkušenosti s vývojem na FPGA čipech firmy Xilinx.

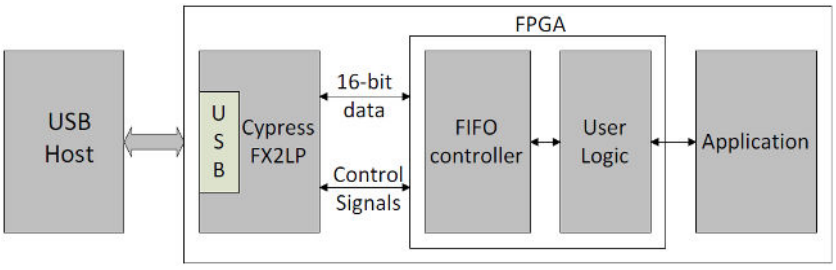
3.3.6 Výběr komunikačního rozhraní mezi hardwarovu a softwarovou částí řídícího systému

S výběrem platformy pro realizaci souvisí i výběr komunikačního rozhraní pro komunikaci mezi počítačem třídy PC a novou řídící jednotkou implementující hardwarovou část řídícího systému. Komunikační rozhraní by mělo být univerzální a podporované systémem MATLAB. Dále musí zajišťovat dostatečnou propustnost a nízkou latenci při komunikaci, aby vykonávání příkazů systému bylo plynulé. Posledním požadavkem je spolehlivost a odolnost.

3.3.6.1 USB – Universal Serial Bus

Prvním z uvažovaných komunikačních rozhraní je USB (univerzální sériová sběrnice), která je velmi dobře podporovaná v každém počítači třídy PC. Při implementaci rozhraní USB pro použití s FPGA je nutné použít dedikovaný čip starající o komunikaci po USB (například obvod Cypress EZ-USB FX2 na obrázku 3.7 [23]), případně použít nakoupený IP Core pro USB nebo si takové IP Core napsat sám. Bohužel napsat takové IP je velmi zdlouhavé díky velké komplikovanosti USB a nákup hotového jádra je poměrně drahou záležitostí.

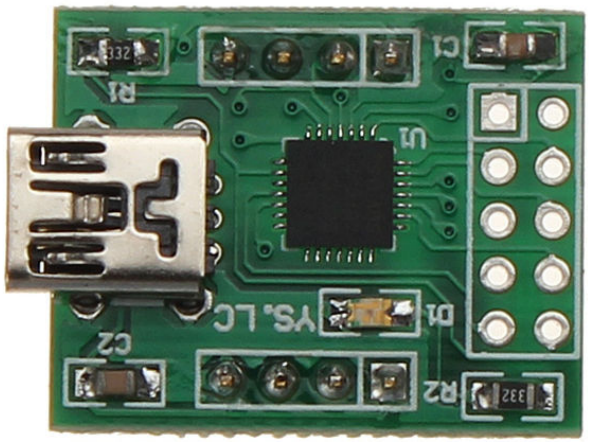
Výhodou je vysoká rychlost přenosu dat (až 480 Mbps). Nevýhodou je pouze polo–duplexní přenos dat, složitá implementace na straně FPGA (obecně hardwarové části), tak na straně softwarového vybavení, kdy je nutné pro takové zařízení napsat ovladač pro každý používaný operační systém. S tím se spojuje pro novější verze operačního systému Windows nutnost takový ovladač certifikovat.



Obrázek 3.7: Architektura a doporučené zapojení obvodu Cypress EZ-USB FX2 a FPGA, převzato z [23]

3.3.6.2 UART — Universal asynchronous receiver/transmitter

Druhým analyzovaným komunikačním rozhraním je asynchronní sériová linka [17]. Jde o nejstarší a nejjednodušší protokol v mém srovnání. Díky prověření časem se používá jako průmyslový standard. Vyznačuje se velkou jednoduchostí, plně duplexní komunikací. V dnešní době jsou u počítačů třídy PC porty pro sériovou linku na ústupu, což se řeší převodníky z USB na UART (příklad takového převodníku je na obrázku 3.8). Typická maximální rychlost komunikace je 115200 baudů (bitů za vteřinu). Tu je však možné navýšit až na 1 Mbps.



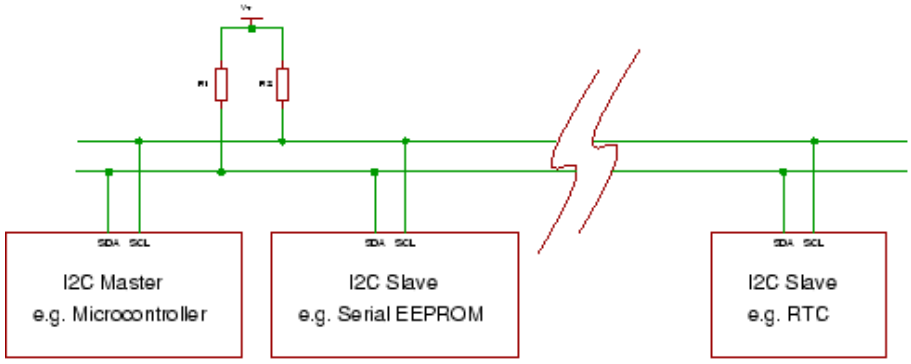
Obrázek 3.8: Příklad průmyslového převodníku USB — UART s čipem CP2102

Výhodou rozhraní UART je snadné a levné zapojení s FPGA, bezproblémová podpora nejrůznějšího softwarového vybavení (bezproblémová podpora v MATLABu bez potřeby externích toolkitů), jednoduchá implementace v hardwaru. Nevýhodou je pouze přenosová rychlost, která by však pro naše využití měla postačovat.

3.3.6.3 I²C — Inter-Integrated Circuit

Sběrnice I²C (Inter-Integrated Circuit) [18] je dvouvodičové datové propojení, využívající signály SDA pro data a SCL pro rozvod hodinového signálu mezi jedním nebo několika obvody typu Master a obvody typu Slave. Všechny součástky jsou připojeny na tutéž sběrnici a jsou cíleně vybírány svými adresami. Adresy i data se přenášejí týmiž vodiči. Sběrnice umožňuje velmi jednoduché propojení mezi několika integrovanými obvody a bezproblémové dodatečné rozšiřování, jak je znázorněno na obrázku 3.9.

Standardní rychlost je 100 kHz a 400 kHz, s příchodem nových verzí rozhraní I²C se rychlost zvýšila až na 3.4 MHz. Výhodou je jednoduchost komunikace a snadná realizace v hardwaru. Komunikační sběrnice I²C je podporována systémem MATLAB prostřednictvím toolkitu. Díky možnosti připojovat více zařízení na jednu sběrnici je zde možnost sledovat informace z připojených senzorů využívají sběrnici I²C. Nevýhodou je však pouze poloduplexní přenos dat a možnost zahltit sběrnici při velkém množství komunikace.



Obrázek 3.9: Typické uspořádání zařízení na sběrnici I^2C, převzato z [5]

3.3.6.4 Vybraná komunikační sběrnice pro komunikace mezi softwarovou a hardwarovou částí řídícího systému

Z výše zmíněných možností pro realizaci komunikace mezi hardwarovou a softwarovou částí řídícího systému jsem se rozhodl pro implementaci rozhraní UART přes USB převodník, jak je popsáno v kapitole 3.3.6.2, z důvodu jeho jednoduché implementaci v hardwarové a softwarové části řídícího systému. Pro naše použití jeho rychlost plně postačuje a jeho výhodou je plně duplexní komunikace umožnující zkrátit reakční doby odezev jednotlivých požadavků na řídící systém.

Kapitola 4

Realizace

Na základě analýzy možných způsobů řešení jsem se rozhodl řídící systém lineárního motoru EZ Limo EZC6E030M–C a jeho řídící jednotkou ESMC–C2 navrhnout a realizovat na FPGA čipu firmy Xilinx. Součástí řídícího systému bude návrh a tvorba desek plošných spojů (DPS) a kromě obslužné aplikace budou vytvořeny i jednoduché demonstrační aplikace v jazyce C/C++ pro ověření konceptu řídícího systému a pro účely testování.

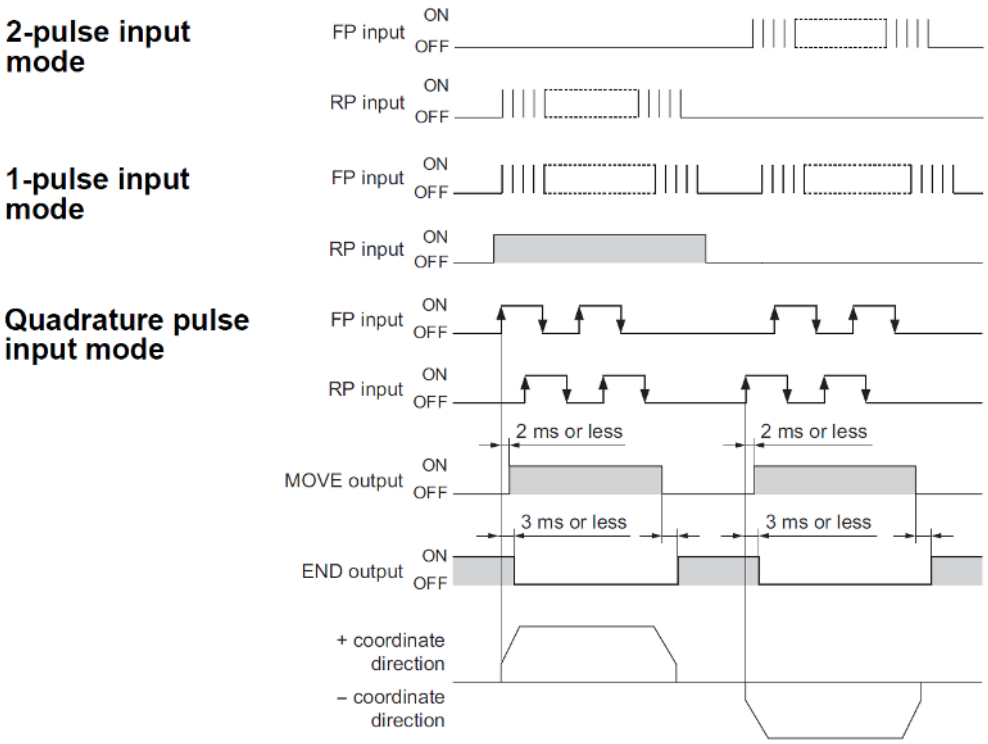
4.1 Řídící jednotka ESMC–C2

Prvním krokem bylo důkladné seznámení se s řídící jednotkou a pochopit principy jejího ovládání v režimu Driver. V manuálu řídící jednotky [2] je podrobný popis všech signálových vstupů a výstupů, včetně příkladů signálových průběhů. Řídící jednotka využívá negativní logiku (klidová úroveň signálů je v logické jedničce — 24 V, aktivní úroveň je v logické nule — 0 V). S výjimkou signálových vstupů pro pohyb vpřed a vzad se funkce řídící jednotky ovládají hladinově. Řídící signály pro pohyb se naopak řídí vzestupnou hranou (v negativní logice tedy sestupnou hranou), kdy motor reaguje pohybem o velikosti minimálního kroku ovládaného motoru.

4.1.1 Časové diagramy nejdůležitějších částí řídící jednotky ESMC–C2

Nejvyšší prioritu pro realizaci funkčnosti řídícího systému je vykonání instrukcí pro pohyb lineárního motoru. Řídící jednotka ESMC–C2 umožňuje ovládat pohyb motoru celkem třemi způsoby. Vybraný způsob komunikace se volí přepínači na řídící jednotce ESMC–C2.

- Řízení každého směru pohybu nezávislým kanálem — Vstupy FP+ a FP- řídící jednotky reagují na sestupnou hranu.
- Řízení multiplexováním vstupu FP+ — Vstup FP+ reaguje na sestupnou hranu. Vstup FP- ovlivňuje směr pohybu motoru.
- Řízení pomocí kvadraturních signálů — Pohyb motoru je řízen nejen na sestupnou hranu signálu, ale i na vzestupnou hranu signálu a zároveň fázovým posuvem mezi hranami signálů. Formát těchto vstupů odpovídá elektromechanickému snímači pohybu motoru.



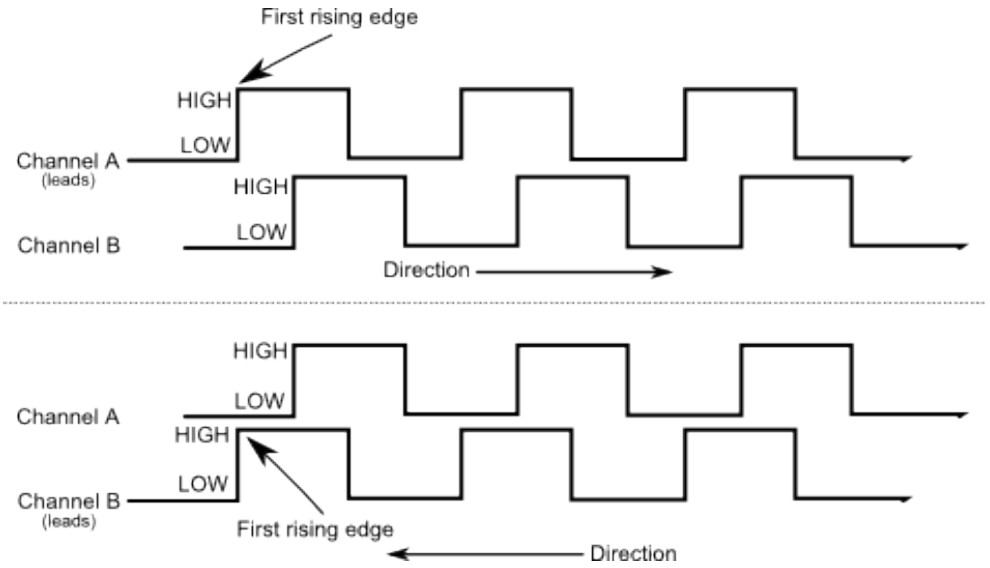
Obrázek 4.1: Časové průběhy signálů pro pohyb motoru pro všechny tři možnosti ovládání

Obrázek 4.1 zobrazuje časové průběhy signálů pro ovládání pohybu motoru. Z tohoto obrázku je patrné, že řídící jednotce a potažmo motoru trvá nezanedbatelný časový interval, než dojde ke zpracování požadavku na pohyb a k jeho skutečnému vykonání. Tomuto jevu se budu věnovat v následující kapitole 4.1.2.

4.1.2 Kvadraturní enkodér

Kvadraturní enkodér je elektromechanický snímač pohybu rotačních částí [7]. Je založen na principu snímání děrovaného kotouče připevněného v ose pohybu motoru, na kterém jsou dvě sekce výřezů, navzájem posunutých o určitou fázi. Díky přidání druhé sekce výřezů můžeme detekovat směr a pohyb. Výsledné hodnoty výstupů (obrázek 4.2) odpovídají v každém kroku Grayově kódu. Grayův kód má tu vlastnost, že při každé změně hodnoty se změní pouze jeden bit ze sledované informace.

V případě řídící jednotky ESMC–C2 jsem se rozhodl prozkoumat, zda by bylo možné použít kvadraturní dekodér jako kontrolu pohybu lineárního motoru v reálném čase. Proto jsem naprogramoval jednoduchý obvod kombinující PWM (Pulse Width Modulation), který vytvářel vždy 1000 impulsů (v případě řídící jednotky ESMC–C2 tato hodnota odpovídá posunu o 10 mm). K němu jsem připojil řídící část, která obsluhovala PWM a přepínala výstupní signál PWM střídavě mezi oba signálové vstupy řídící jednotky určené pro pohyb motoru vpřed a vzad.



Obrázek 4.2: Výsledný digitální signál z kvadraturního enkodéru, převzato z [7]

Na obrázku 4.3 je zobrazen výstup z kvadraturního enkodéru (spodní 2 průběhy) spolu se signály pro pohyb lineárního motoru (horní 2 průběhy). Z obrázku je jasně patrný fázový posuv mezi budícími signály a skutečným pohybem motoru, v tomto případě více než 10 ms.

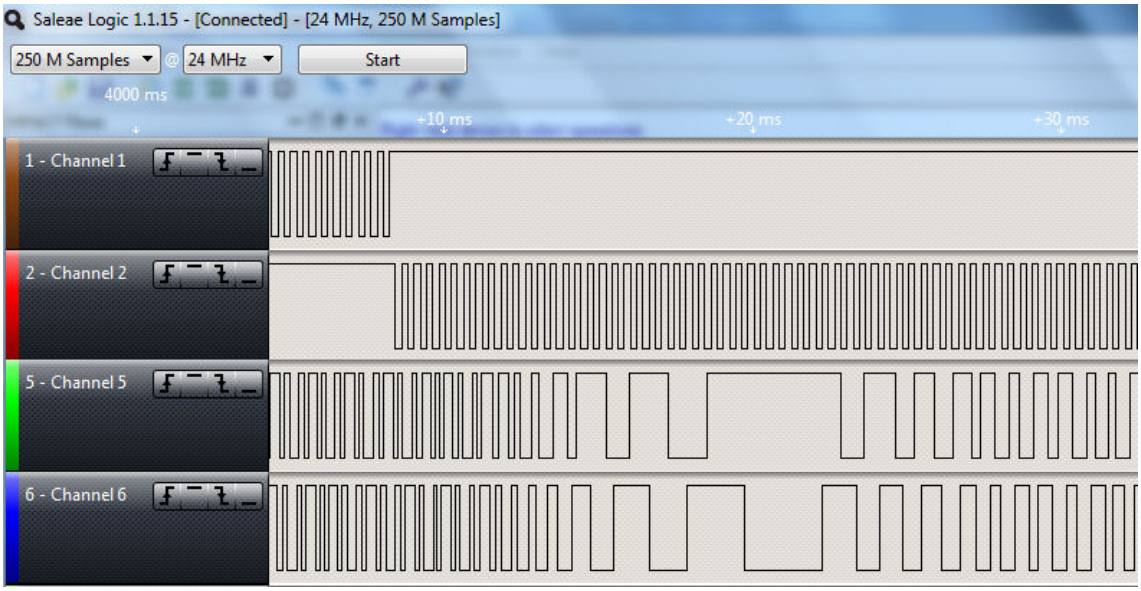
Takový čas je vzhledem k typické taktovací frekvenci čipů FPGA (50 MHz a více) příliš vysoký, aby mohl sloužit k ověřování polohy v reálném čase. Z obrázku zároveň patrná akcelerace a decelerace lineárního motoru, kterou si kontroluje řídící jednotka ESMC–C2 sama.

Další částí bylo ověření, zda lineární motor a jeho řídící jednotka dokážou i přes opakovaný pohyb návrat do stejné polohy. Pro tento úkol jsem rozšířil program pro generování PWM, ke kterému jsem přidal možnost přidat možnost ručně nastavovat směr posunu, velikosti posuvu (1 a 1000 kroků) a resetovací vstup pro vynulování hodnoty čítače pozice, kterou jsem dekódoval z kvadraturního enkodéru.

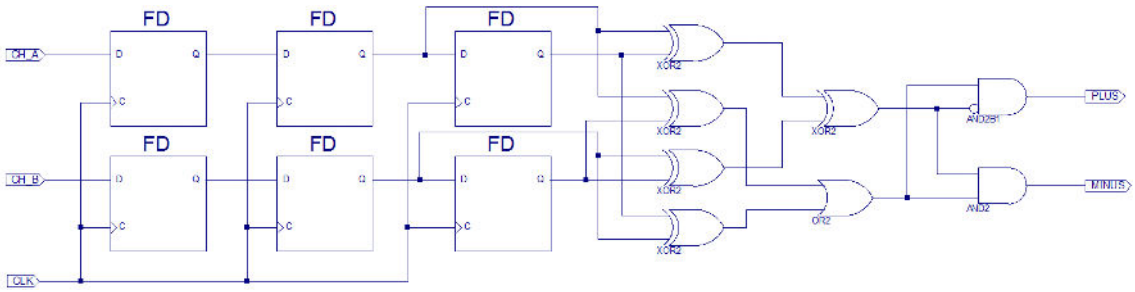
Pro dekódování signálu z kvadraturního enkodéru jsem použil zapojení na obrázku 4.4. Z obrázku jsou patrné metastabilní filtry pro synchronizaci přijímaného signálu (dva sériově řazené klopné obvody typu D). Prostřední část je věnována 4 hranovým detektorům. Výstup hradla OR je signál „směr“, výstupem posledního hradla XOR je signál „změna“. Poslední část obvodu je jednobitový dekodér, který signály „směr“ a „změna“ převádí na signály PLUS a MINUS, které slouží pro inkrementování (dekrementování) hodnoty čítače pozice. Obsah čítače pozice je poté vyveden na LED diody (spodních 10 bitů).

Při několik testech na tomto obvodu jsem došel k závěru, že jeden puls buzení lineárního motoru odpovídá 4 inkrementacím (dekrementacím) čítače pozice. To vyplývá ze skutečnosti, že pohyblivá část motoru vykoná jednu celou obrátku, sledovací senzor kvadraturního enkodéru má však rozlišení na jednu čtvrtobrátku. To je patrné i na obrázku 4.1, kdy způsob ovládání motoru pomocí kvadraturních vstupů vyžaduje 4x více pulsů než zbývající metody.

Z tohoto důvodu bude v řídícím systému čítač pro snímání reálné pozice získávané z kvad-



Obrázek 4.3: Výsledný digitální signál z kvadraturního enkodéru – zachycený logickým analyzátorem



Obrázek 4.4: Použité zapojení pro kvadraturní dekodér založené na [3]

raturního enkodéru větší o 2 bity, které budou při vysílání stavových informací zanedbávány. Výsledná hodnota je tedy zaokrouhlena na nejbližší násobek 4.

Posledním důležitým poznatkem je drobná nepřesnost při změně směru pohybu motoru, kdy není vykonána celá obrátka pohyblivé části. Výsledná hodnota při změně směru se liší typicky o jeden puls kvadraturního enkodéru. Při zaokrouhlování tuto vlastnost nemusím příliš řešit. To odpovídá i parametrům udávaným výrobcem, kdy chyba při pokusu o opakované dosažení výchozí polohy by měla být ±2 kroky lineárního motoru (±0.02 mm).

4.1.3 Ovládání magnetické brzdy a násilné zastavení motoru

Lineární motor EZ Limo EZC6E030M–C má v sobě zabudovanou magnetickou brzdu, kterou lze v případě aktivace manipulovat pohyblivou částí motoru jiným způsobem, než signálovými vstupy pro pohyb motoru. Je tedy možné s pohyblivou částí manipulovat rukou napří-

klad pro prvotní nastavení výchozí polohy. Spolu se signálovým vstupem C.OFF (Current Off) magnetická brzda odpojí přívod elektrického proudu do lineárního motoru. Funkci násilného zastavení motoru můžeme realizovat magnetickou brzdou i funkcí C.OFF.

V případě využití funkce C.OFF máme jistotu, že se pohyblivá část motoru nepohne například vlivem zavěšeného těžkého tělesa. Funkce C.OFF je jinak stejná jako funkce magnetické brzdy. Přesto jsem si pro realizaci vybral implementaci pouze magnetickou brzdu. Funkci násilného zastavení motoru lze realizovat uvedením signálových vstupů pro pohyb motoru do klidových úrovní.

4.1.4 Elektrické zapojení mezi řídící jednotkou ESMC–C2 a FPGA

Protože výše popsané funkce řídící jednotky ESMC–C2 dostačují k realizaci řídícího systému a ostatní v analýze popsané signálové vstupy a výstupy lze považovat za podpůrné. Jejich implementace přijde na řadu až po odladění velké části řídícího systému, aby nakonec bylo možné sledovat co nejvíce informací, které poskytuje řídící jednotka ESMC–C2.

V průběhu vývoje a testování se vyskytly problémy u některých zapojení, proto uvedu všechny hlavní zapojení i s případnými neduhy.

4.1.4.1 Elektrické zapojení číslo jedna

Při návrhu první verze elektrického zapojení (obrázek 4.5) jsem neměl dostatek praktických zkušeností a výsledkem bylo zapojení, které sice fungovalo, ale trpělo přehříváním použitého optočlenu. Toto zapojení neinvertuje, takže se lépe sledovaly generované průběhy signálů na osciloskopu nebo logickém analyzátoru.

Protože řídící jednotka ESMC–C2 má jako klidovou úroveň +24 V, je optočlen po většinu času sepnutý. Každou částí optočlenu při sepnutí protékalo přibližně 20 mA, což je příčinou zvýšené teploty optočlenu. Tuto teplotu jsem nepovažoval za vhodnou pro dlouhodobý provoz.

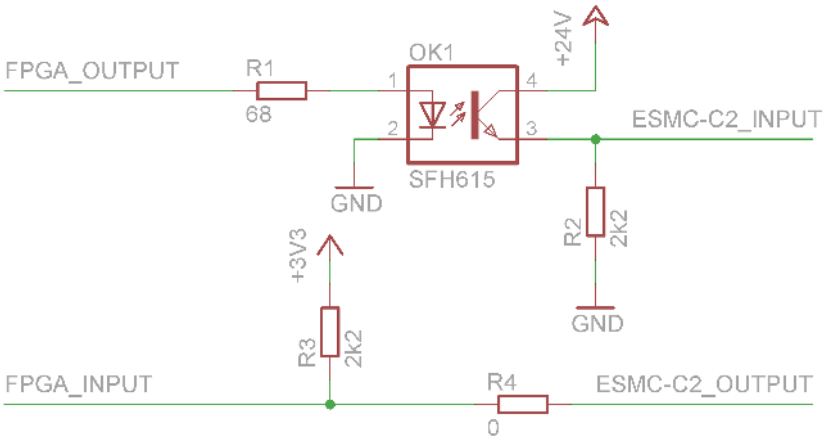
Vstup do FPGA je řešen pouze výtažnými (pull-up) rezistory. Toto zapojení nelze použít pro galvanické oddělení obvodů.

4.1.4.2 Elektrické zapojení číslo dva

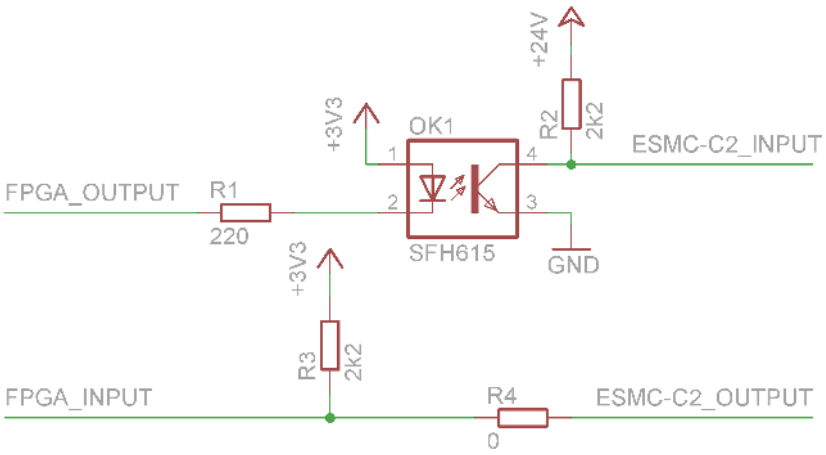
Druhá verze elektrického zapojení (obrázek 4.6) již netrpí přehříváním optočlenu, ale stále se nedá použít pro galvanické oddělení původní řídící jednotky ESMC–C2 a nové řídící jednotky realizované v FPGA. Vstup do FPGA je řešen stejně jako v předchozím zapojení, tedy pouze výtažnými (pull-up) rezistory. Protože je jednotka ESMC–C2 ovládána nikoliv napěťovou úrovní, ale proudovou smyčkou, výtažný rezistor za optočlenem je zbytečný a pouze navyšuje spotřebu celého zapojení.

4.1.4.3 Elektrické zapojení číslo tři — finální verze pro realizaci na DPS

Třetí a finální verze elektrického zapojení (obrázek 4.7) je uzpůsobena pro galvanické oddělení obou částí systému. Jsou zde použity vysokorychlostní optočleny s integrovaným operačním



Obrázek 4.5: První verze elektrického zapojení

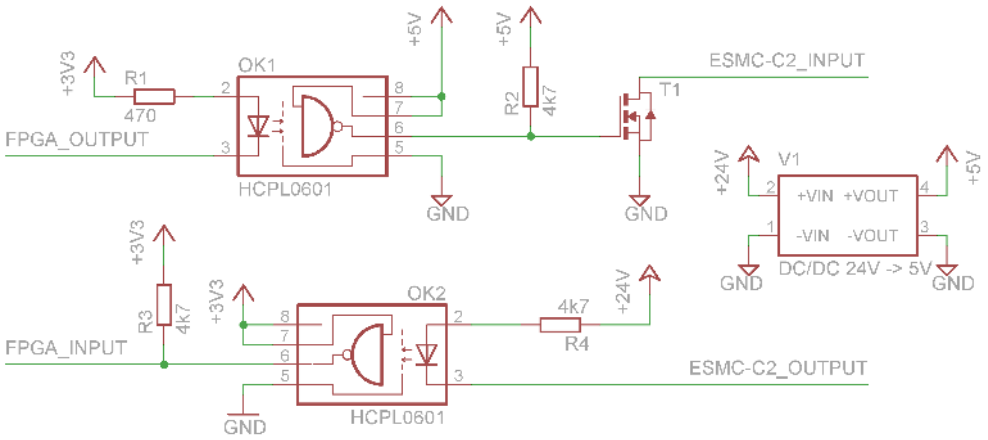


Obrázek 4.6: Druhá verze elektrického zapojení

zesilovačem kombinované s fotodiodou. Díky tomu lze optočlen budit velmi malým proudem do LED diody, což se pozitivně projeví na spotřebě optočlenů.

Vysokorychlostní optočleny jsem se rozhodl použít na základě lepšího tvaru výsledného digitálního signálu, který při vyšších frekvencích je mnohem lépe zpracováván řídící jednotkou ESMC–C2 a nedochází tak na vyšších rychlostech k zatuhnutí řídící jednotky. Nevýhodou je maximální napájecí obvodu +5 V, což se projevilo nutností přidat DC/DC měnič pro tvorbu potřebného napětí +5 V.

Toto zapojení už podporuje galvanické oddělení, neboť je zde pro každou stranu separátní svorka s úrovní 0 V (zem). Nevýhodou zapojení je invertování signálů, proto je s tím nutné počítat na straně FPGA oproti předchozím zapojením.



Obrázek 4.7: Třetí a finální verze elektrického zapojení

4.1.5 Ostatní signálové vstupy a výstupy řídící jednotky ESMC–C2

Všechny ostatní nezmiňované signálové vstupy z tabulek 2.3 a 2.2 mají podpůrný charakter v rámci řídící jednotky ESMC–C2. Jejich funkcionalita není kritická, přesto jsou k nim na desce plošných spojů (DPS) vytvořeny stejná elektrická zapojení, jako pro nejpoužívanější signálové vstupy a výstupy. V případě prohlášení DPS za finální verzi se veškeré úpravy řídícího systému budou provádět jen v rámci FPGA a nebude nutné vyvíjet další DPS s doplněnou funkcionalitou.

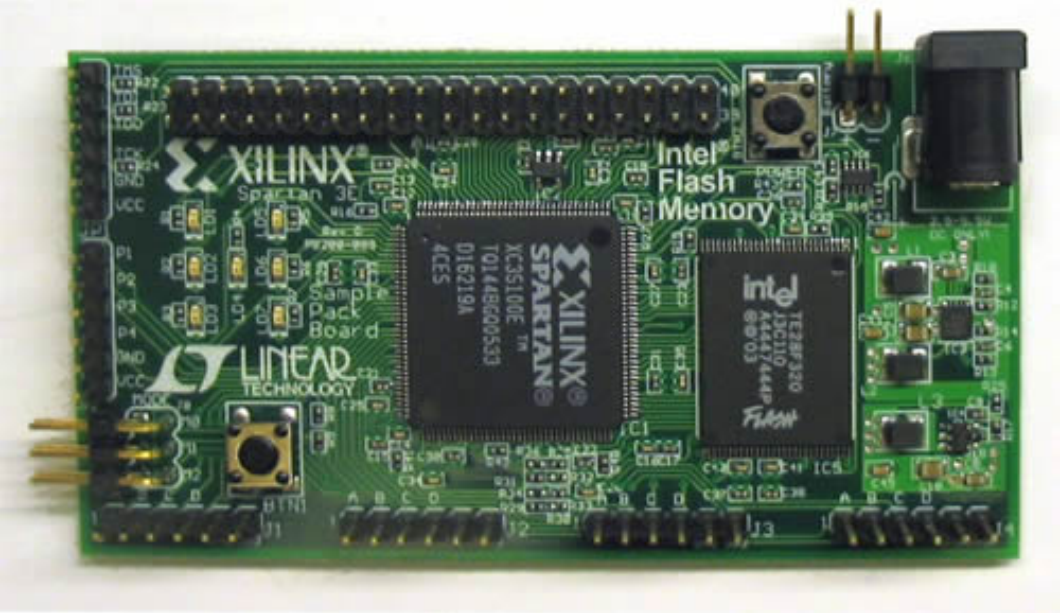
4.1.6 Doporučené parametry kladené na použitou kabeláž

Výrobce řídící jednotky ESMC–C2 doporučuje použití stíněné kabeláže pro všechny signálové i napájecí rozvody. To by mělo zvýšit odolnost celého řídícího systému vůči elektromagnetickému rušení a splnit tak požadavky kladené na elektromagnetickou kompatibilitu (EMC).

V prvotní fázi realizace jsem z důvodu jednoduchosti zapojení a montáže použil obyčejné nestíněné vodiče. To se mi hrubým způsobem vymstilo, jak bude později podrobněji popsáno v kapitole testování.

4.2 Návrh části řídícího systému realizované v FPGA

Druhým krokem v návrhu a realizaci řídícího systému byl detailní rozbor dostupného čipu Xilinx Spartan–3E 100k, který jsem měl k dispozici na vývojové desce Xilinx Spartan–3E Sample Pack [11] (obrázek 4.8). Tuto vývojovou desku jsem zakoupil pro své experimenty s FPGA, nebylo tedy nutné řešit zapůjčení vývojové desky z jiného zdroje. Použitý obvod Xilinx XC3S100E je nejmenší v rámci rodiny Spartan–3E, která je navržena pro největší hustotu logických funkcí. Proto je důležité se pokusit využít co nejefektivněji dostupné zdroje tohoto čipu FPGA. Předpokládal jsem, že při efektivním využitím zdrojů poskytovaných čipem bude jeho velikost dostatečná pro realizaci řídícího systému.



Obrázek 4.8: Vývojová deska Xilinx Spartan–3E Sample Pack, převzato z [11]

4.2.1 Vývojová deska Xilinx Spartan–3E Sample Pack

Vývojová deska Xilinx Spartan–3E Sample Pack (obrázek 4.8) měla v době svého uvedení demonstrovat možnosti tehdy nejpokročilejšího FPGA čipu firmy Xilinx pro spotřební elektroniku. Kromě čipu Xilinx Spartan–3E 100K obsahuje i několik DC/DC měničů pro zajištění napájení čipu a paralelní Flash paměť firmy Intel pro uložení konfigurace FPGA čipu a případné uložení uživatelských dat.

4.2.2 Části přímo ovlivněné architekturou čipu Spartan–3E

Jak je patrné z tabulek 4.1 a 4.2, obvod Spartan–3E o velikosti 100K ekvivalentních hradel není příliš velký. Z pohledu implementace je nejdůležitější navrhnout systém plynulého zásobování systému instrukcemi, které mohou mít různé priority. K tomu se nejlépe hodí implementace fronty na čipu FPGA.

Vzhledem k analýze a vybrání datového typu Integer jako formát pro přenos informací o standardní délce 32–bitů jsem se rozhodl použít vestavěné celky paměti RAM — BlockRAM. BlockRAM jsou části čipu FPGA, které není možné programovat do libovolné struktury, neboť jsou optimalizovány jako paměťové buňky pro vytvoření paměti typu RAM.

4.2.2.1 Způsob kódování instrukcí

Při implementaci jsem se musel rozhodnout mezi dvěma způsoby realizace paměti instrukcí: zakódovat instrukce pevnou délkou instrukčního slova nebo použít kódování s proměnnou délkou instrukce. Výhodou pevné délky instrukčního slova je jednoduchost adresního řadiče a

Dostupné zdroje v čipu Xilinx XC3S100E–TQ144–4C ES	
Maximální počet I/O portů	108
Počet Sliců	960
Počet LUTů	1920
Počet klopných obvodů	1920
Velikost BlockRAM	72 kbit
Počet BlockRAM	4
Digital Clock Managment (DCM)	2
Dedikované násobičky 18x18	4
Rychlost obvodu	-4 (standardní)
Typ obvodu	pro běžné použití
Inženýrský vzorek (ES)	Ano

Tabulka 4.1: Dostupné zdroje v čipu Xilinx XC3S100E–TQ144–4C ES, převzato z [10]

Použité součástky na vývojové desce	
XC3S100E-TQ144-4	Popis FPGA 4.1
Intel StrataFlash 28F320J3	32 Mbit parallel flash NOR memory
LTC6905-50	Oscilátor o frekvenci 50 MHz
LED Diody	7 kusů uspořádané do písmena H
Tlačítko	1 kus, doporučeno pro funkci reset
Napájení	DC Jack a header pro baterii, zapínání přes tlačítko
Počet vyvedených I/O pinů	52

Tabulka 4.2: Použité součástky na vývojové desce, převzato z [11]

jeho realizace. Nevýhodou je plýtvání místem v paměti RAM. Výhodou kódování proměnnou délkou je úspora paměti, nevýhodou je složitost adresního řadiče.

Protože při samotném běhu řídícího systému se budou z velké části provádět jen příkazy pro pohyb motoru, bude většina instrukcí dosahovat maximální délky instrukčního znaku. Prioritní instrukce nenesou žádné jiné informace než svůj kód a jsou krátké. Jejich ukládání se však bude dít do paměti pro prioritní instrukce, kterou můžu volit záměrně malou.

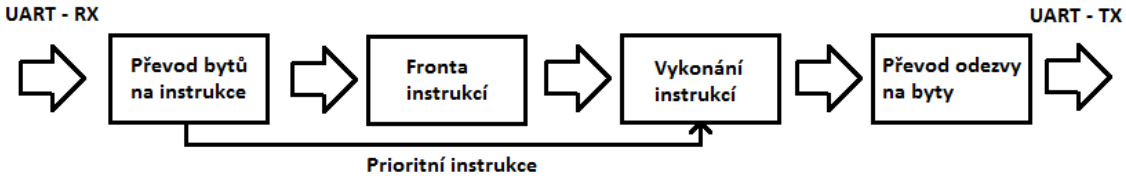
Pro realizaci řídícího systému jsem se tak rozhodl pro použití kódování instrukcí, které bude mít fixní délku. V tomto případě bude nevyužité paměti jen velmi málo a jednoduchý adresní řadič je také přínosem.

4.2.2.2 Paměť pro uložení instrukcí

Při realizaci paměti pro uložení instrukcí jsem vybíral mezi dvěma programátorskými modely paměti. Jedním z nich je klasická RAM a zpracování instrukcí bude velmi podobné zpracování programu procesorem.

Druhým modelem je jednoduchý konečný automat (FSM), který bude zpracovávat instrukci za instrukcí, které bude získávat ze vstupní fronty instrukcí, která bude zároveň

sloužit jako vyrovnávací paměť. Na vstupní části fronty bude připojena část, která bude transformovat byty přicházející po sériové lince na jednotlivé instrukce. Tento model jsem si zároveň vybral pro realizaci řídícího systému, protože přirozeným způsobem implikuje činnost hardwarové části řídícího systému. Přibližný model je na obrázku 4.9.



Obrázek 4.9: Koncepce hardwarové části řídícího systému

4.2.2.3 Realizace fronty na čipu Spartan–3E

Pro realizaci fronty jsem se rozhodl použít vestavěnou paměť RAM — BlockRAM. Ta má velikost 18 kbit a umožňuje velké množství rozložení paměťových buněk dovolující co nejoptimálnější využití BlockRAM pro data s různou šířkou. BlockRAM umožňuje použít dva nezávislé porty pro přístup do paměti RAM.

Možné konfigurace paměťového bloku Block RAM	
Šírka ukládaných dat	Počet možných záznamů
1 bit	16384
2 bity	8192
4 bity	4096
8 bitů	2048
9 bitů (8 bitů + paritní bit)	2048
16 bitů	1024
18 bitů (16 bitů + 2×paritní bit)	1024
32 bitů	512
36 bitů (32 bitů + 4×paritní bit)	512
72 bitů	256 (pouze jednoportová RAM)

Tabulka 4.3: Možné konfigurace paměťového bloku BlockRAM u čipu Spartan–3E, převzato z [10]

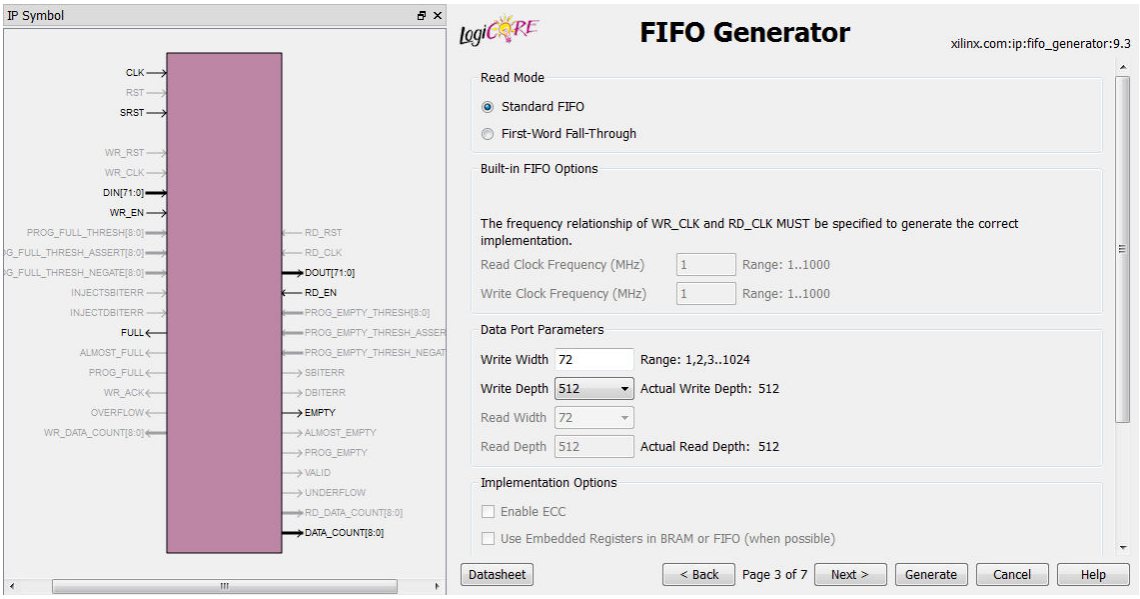
Jak je patrné z tabulky 4.3, nejširší šířku dat umožňuje konfigurace 36 bitů při 512 záznamech, která ještě dovoluje použít oba porty BlockRAM. Jeden z těchto portů bude použit jako čistě zápisový port (vstupní strana fronty) a druhý bude použit jako čistě pro čtení dat ze fronty.

Na základě analýzy (kapitola 3.2.2.2) jsem jako formát dat použil datový typ Integer o velikosti 32 bitů. Tato šířka dat se tedy bez problémů vejde do jedné BlockRAM. Největší příkaz je příkaz MOVE (pohyb motoru o udanou vzdálenost a čas). Tento příkaz má dva

parametry, každý o velikosti 32 bitů. Z tohoto důvodu je nutné spojit paralelně dvě dílčí fronty pro 36 bitů. Šířka dat 36 bitů znamená optimálně využít paměťovou konfiguraci BlockRAM. Spojením dvou front tedy získám frontu o šířce dat 72 bitů a velikosti 512 záznamů.

Nutná šířka dat pro dvě čísla typu Integer je 64 bitů, tedy zbývá ještě 8 využitelných bitů. Tyto bity využiji pro uložení hlavičky instrukce. V 8 bitech jsem schopen vyjádřit 256 možných hodnot, tedy i 256 instrukcí. To by mělo pro realizaci řídícího systému postačovat. Další výhodou 72–bitové šířky dat je dělitelnost 8 bity, tedy velikosti přenášených dat přes rozhraní UART. Tedy budu optimálně využívat i přenosové pásmo sériové linky.

Z vyvozených parametrů fronty můžu přistoupit k samotné realizaci fronty. Rozhodl jsem se použít výrobcem dodávaný automatizovaný generátor front [6] (obrázek 4.10). Tím jsem si ušetřil čas při návrhu vlastní fronty s jistotou důkladného otestování vygenerované fronty výrobcem čipu firmou Xilinx.



Obrázek 4.10: Automatizovaný generátor front firmy Xilinx, převzato z [6]

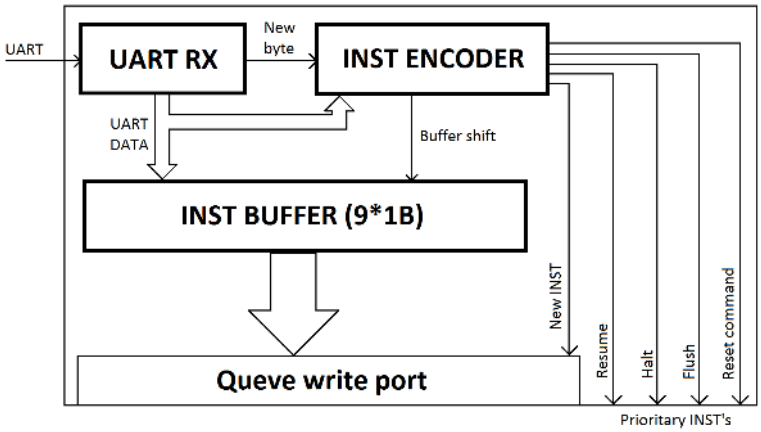
Výsledný návrh skutečně zabírá 2 kusy BlockRAM a zanedbatelné množství logiky a klopných obvodů pro adresní řadič fronty. Výhodou použití generátoru je možnost v případě potřeby kdykoliv změnit parametry fronty téměř bez nutnosti úprav návrhu.

4.2.3 Zakódování a formát instrukcí nové řídící jednotky

V předchozích kapitolách jsem se rozhodl pro použití instrukcí s pevnou šířkou slova 72 bitů, kde 8 bitů je určeno pro zakódování hlavičky instrukce. Zbylých 64 bitů bude rozděleno na 2 × 32 bitů pro reprezentaci celočíselných operandů instrukcí. Při kódování jsem použil kombinace prefixového a lineárního kódu. Tabulka B.1 obsahuje kompletní přehled zakódovaných instrukcí.

4.2.4 Modul pro příjem instrukcí přes sériovou linku

Vstupní částí řídícího systému realizovaném v hardwaru je modul pro příjem instrukcí přes rozhraní sériové linky (UART). Součástí tohoto modulu bude i podmodul starající se o příjem bytů. Z těchto bytů budou zformovány zprávy s instrukcemi, které budou vloženy do fronty prostřednictvím zápisového portu.



Obrázek 4.11: Modul pro příjem instrukcí s nejdůležitějšími částmi

Na obrázku 4.11 jsou znázorněny nejdůležitější části modulu pro příjem instrukcí. Podmodul pro příjem bytů z rozhraní UART přijímá znaky a informuje nadřízený modul INST ENCODER. Rychlost přenosu dat je nastaven na 115200 baudů, s možností změny pomocí generického parametru. Ostatní parametry jsou fixní a jsou uvedeny v tabulce 4.4. Přenos dat nevyužívá jakýkoliv způsob kódování, například samoopravným nebo detekčním kódem.

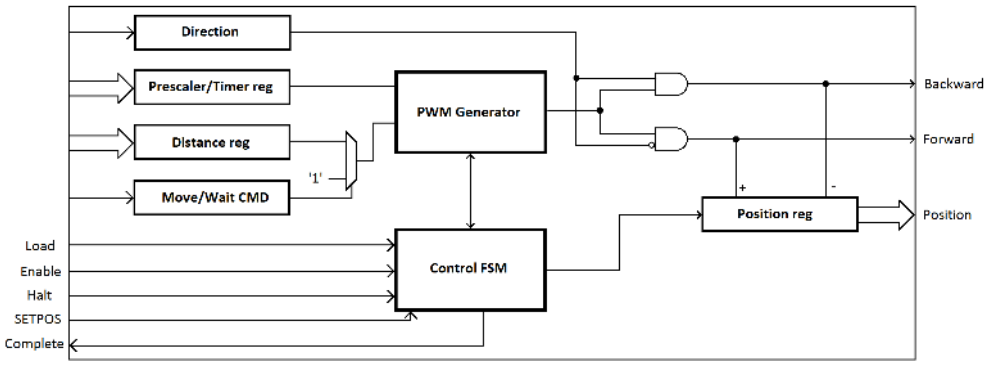
Parametry pro přenos dat přes rozhraní UART	
Parametr	Hodnota
Přenosová rychlost	115200 baudů
Počet bitů	8
Počet stop bitů	1
Parita	Ne
Řízení toku dat	Ne

Tabulka 4.4: Parametry pro přenos dat přes rozhraní UART

Podmodul INST ENCODER je nadřízený podmodulu UART RX, který rozpoznává na základě přijímaných dat typ instrukce a její délku. Pokud přijme data o stejné velikosti jako je očekávaná délka instrukce, je proveden zápis instrukce do instrukční fronty s výjimkou prioritních instrukcí. Vzhledem k jejich jednoduchosti realizace v hardwaru budou tyto instrukce vždy realizovány rychleji než příjem dalšího bytu dat přes rozhraní UART. Proto jsou prioritní instrukce řešeny samostatnými signály, které obcházejí instrukční frontu a vedou rovnou do modulu pro zpracování instrukcí.

4.2.5 Modul pro generování pulsů pro buzení motoru

Schéma na obrázku 4.12 popisuje funkční blok pro generování obdélníkových pulsů, které budou sloužit pro buzení motoru prostřednictvím řídící jednotky ESMC–C2. Tento funkční blok obsahuje pulsně – šířkový generátor (PWM), který na základě registrů Prescaler (32–bitový) a Distance (32–bitový) vygeneruje příslušný počet obdélníkových pulsů se zadaným časovým rozestupem. Druhou funkcí je vytváření prodlevy mezi jednotlivými příkazy. Doba prodlevy se nastavuje registrem Timer (32–bitový).



Obrázek 4.12: Modul generování pulsů pro buzení lineárního motoru

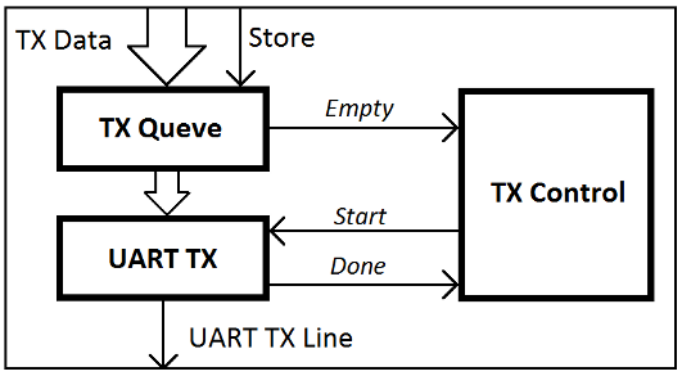
Výběr použitého režimu se provádí přes registr Move/Wait CMD. Společné signály pro celou jednotku jsou signály:

- Load – Načtení hodnot do registrů.
- Enable – Povoluje generování pulsů na výstupu. Umožňuje znovuobnovení činnosti.
- Halt – Násilné zastavení příkazu bez možnosti obnovy činnosti.
- Complete – Oznamuje nadřízenému bloku vykonání požadovaného příkazu.

Posledním výstupem z jednotky obsah registru Position, který udržuje hodnotu pozice motoru. Tato pozice není garantovaná a je určena pro vnitřní účely řídícího systému. Obsah registru se inkrementuje/dekrementuje s každým vygenerovaným pulsem. V řídícím systému slouží pro kontrolu pozice motoru vůči dorazům. Tento registr je možné nastavit na uživatelem definovanou hodnotu signálem SETPOS.

4.2.6 Modul pro vysílání odezvy řídícího systému pře sériovou linku

Pro získávání informací ze systému prostřednictvím sériové linky – UART je nutné zajistit nezávislé vysílání vyžádaných informací a garantovat jejich správné pořadí. Zároveň je nutné, aby obsluha vysílání co nejméně zatěžovala a zpomalovala činnost samotného řídícího systému. Z tohoto důvodu jsem se rozhodl použít generátoru front dodávaného výrobcem FPGA popsaného v kapitole 4.2.2.3. Tím je možné redukovat čas strávený ke komunikaci mezi nadřazeným blokem vykonávajícím instrukce a modulem pro vysílání odezvy. Vnitřní zapojení je patrné z obrázku 4.13.



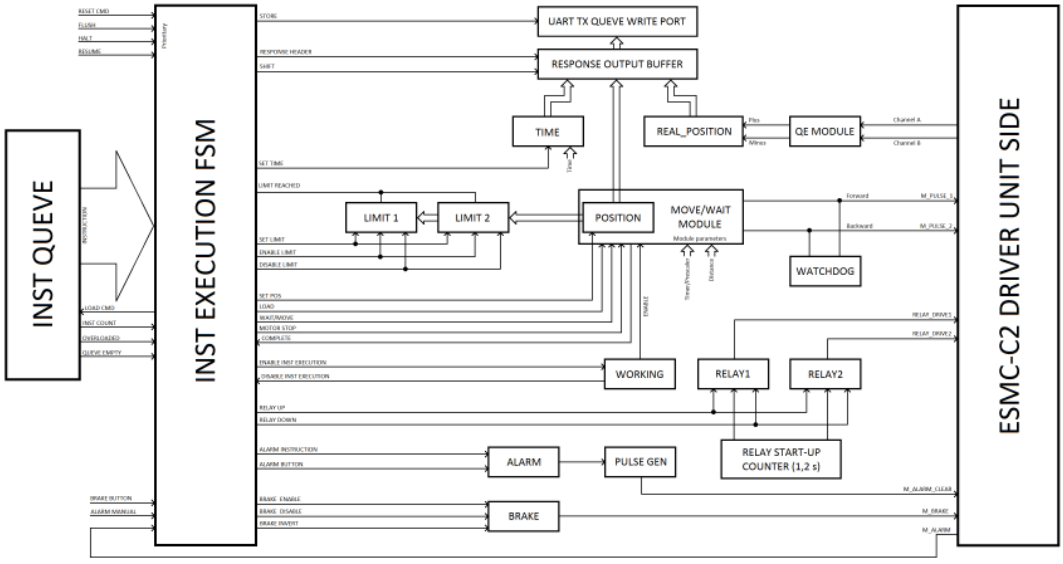
Obrázek 4.13: Modul pro vysílání dat přes sériovou linku – UART

4.2.7 Hlavní modul pro zpracování instrukcí a řízení jednotky ESMC–C2

Nejdůležitější částí hardwarové části řídícího systému je modul pro zpracování instrukcí (tabulka s instrukcemi B.1). Pro některé instrukce je vyžadováno vygenerování odezvy, která je vysílána prostřednictvím modulu pro vysílání přes sériovou linku (kapitola 4.2.6). Protože jde o velmi rozsáhlý modul, je jeho slovní popis a přístup k realizaci rozdělen na další pododdíly.

4.2.7.1 Schéma vnitřního zapojení hlavního modulu

Na obrázku 4.14 je znázorněno vnitřní zapojení modulu pro zpracovávání instrukcí řídícího systému. Větší obrázek zapojení je pro lepší čitelnost v příloze C.1.



Obrázek 4.14: Vnitřní zapojení hlavního modulu pro provádění instrukcí

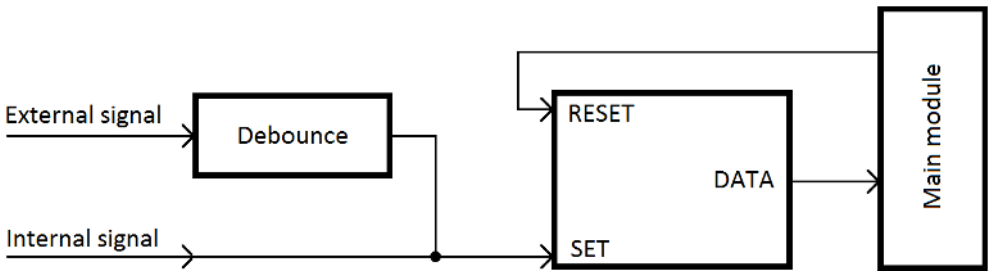
4.2.7.2 Seznam vstupních a výstupních portů

- clk – rozvod systémových hodin
- reset – rozvod resetovacího signálu
- Obsluha fronty příkazů:
  - cmd – instrukce o šířce 72 bitů
  - cmd_count – počet instrukcí zbývajících ve frontě
  - q_empty – fronta příkazů je prázdná
  - load_cmd – načti instrukci z fronty instrukcí
  - flush_queve – smaž celý obsah fronty instrukcí
- Výstupní port pro vysílací modul:
  - tx_data – jeden byte pro vysílání
  - tx_store – zapsání bytu do fronty pro vysílání
- Vstupně – výstupní signály pro řídící jednotku ESMC–C2:
  - Pohyb motoru:
    - m_pulse_1 – pohyb motoru vpřed
    - m_pulse_2 – pohyb motoru vzad
  - m_brake – ovládání magnetické brzdy
  - Kvadraturní enkodér
    - m_qe1 – kanál 1 kvadraturního enkodéru
    - m_qe2 – kanál 2 kvadraturního enkodéru
- Vstupy prioritních instrukcí:
  - halt_cmd – signál pro přerušení činnosti řídícího systému
  - resume_cmd – signál pro zahájení/obnovení činnosti řídícího systému
  - flush_cmd – signál pro zastavení činnosti řídícího systému a vymazání instrukční fronty
- Stavové informace:
  - working_flag – informace o činnosti řídícího systému
  - limit1_reached – dosažení hraničního bodu č.1
  - limit2_reached – dosažení hraničního bodu č.2
- Reléové výstupy:
  - relay_drive1 – reléový výstup č.1
  - relay_drive2 – reléový výstup č.2

- Obsluha alarmu na řídící jednotce ESMC–C2:
  - m_alarm – Výstup ESMC–C2 označující chybu
  - m_alarm_btn – Možné manuální vymazání alarmu
  - m_alarm_clear – Vymazání alarmu pomocí instrukce řídícího systému
- Ostatní vstupy:
  - brake_switch – manuální zapnutí/vypnutí magnetické brzdy
  - queve_overloaded – některá z front je přeplněná – možná chyba řídícího systému

4.2.7.3 Způsob předávání informací uvnitř a vně hardwarové části řídícího systému

Pro předávání informací uvnitř a vně systému bylo nutné navrhnout a realizovat protokoly, které zajistí správnost přenášených dat. Pro komunikaci mezi jednotlivými moduly jsem zvolil předávání řídících signálů pomocí Set–Reset klopných obvodů. Zapojení je na obrázku 4.15.



Obrázek 4.15: Zapojení protokolu pro komunikaci mezi moduly

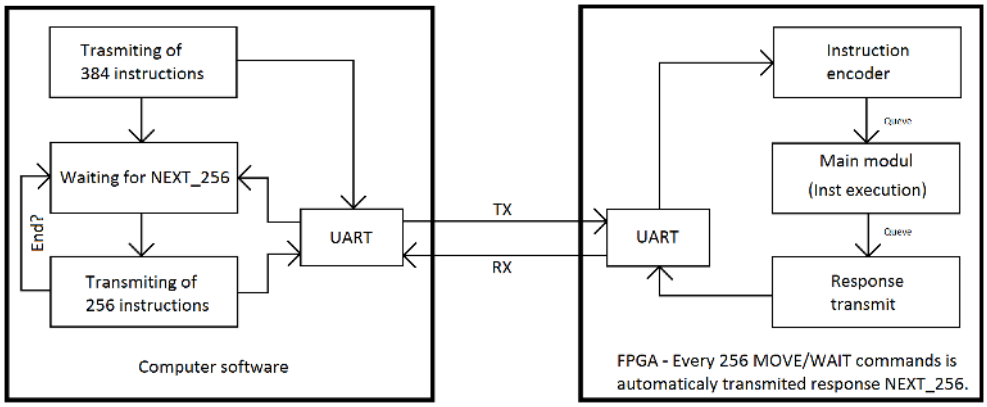
Toto zapojení je řešením problému producent–konzument pro hardware za předpokladu jednoho konzumenta. Je možné připojit více konzumentů, ale poté nejsme schopni rozlišit zdroj události ani počet výskytů od poslední obsluhy. To jsem řešil návrhem stavového automatu, který potenciálně kritické řídící signály zpracovává s vyšší prioritou. To konceptuálně odpovídá přerušovacímu systému v počítačích s přiřazením priority pro každý zdroj přerušení.

Pokud řídící signál přichází z vnějšího prostředí (tlačítko pro obsluhu), je řídící signál nejdříve zpracován zákmitovým filtrem.

4.2.7.4 Komunikační protokol mezi softwarovou a hardwarovou částí řídícího systému

Komunikační protokol pro komunikaci mezi softwarovou a hardwarovou částí řídícího systému využívá pro samotný přenos dat sériovou linku – UART. Toto komunikační rozhraní jsem popsal v kapitole 3.3.6.2. Schéma komunikačního protokolu je na obrázku 4.16.

Protokol vychází z poznatků při návrhu a realizaci fronty příkazů 4.2.2.3 a šířky instrukčního slova 4.2.3. Základní velikost bloku pro komunikaci je 256 instrukcí (polovina



Obrázek 4.16: Protokol pro komunikaci mezi hardwarovou a softwarovou částí řídícího systému

velikosti instrukční fronty) typu MOVE/WAIT. V hardwarové části řídícího systému je realizován čítač instrukcí MOVE/WAIT, který vyšle zprávu vždy po vykonání 256 instrukcí MOVE/WAIT. Velikost prvního vysílaného bloku je nastavena na 384 (1.5–násobek základní velikosti bloku). Tím je zajištěna plynulost celého pohybu motoru, protože v okamžiku vyslání by ve frontě instrukcí mělo zbývat ještě 128 instrukcí.

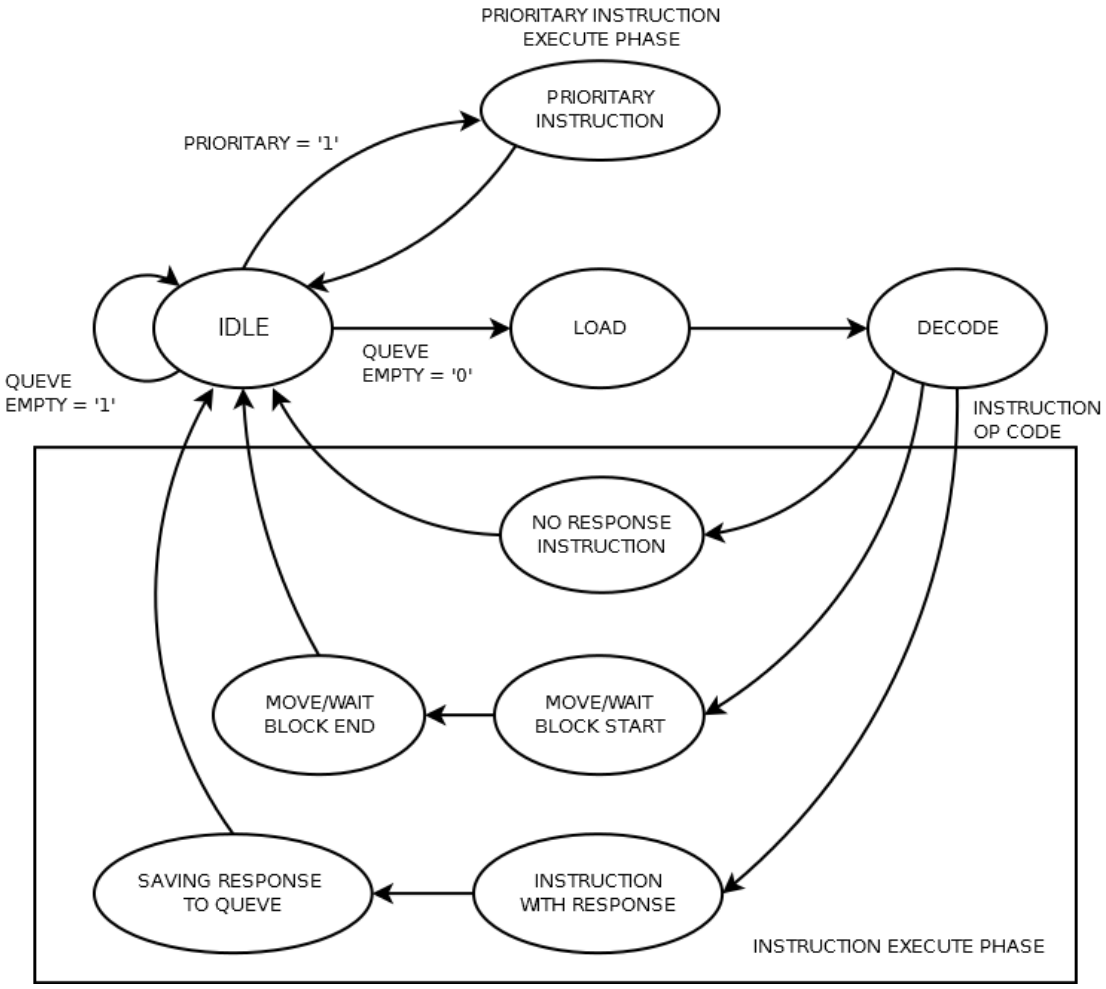
Ostatní instrukce se do velikosti bloku nezapočítávají vzhledem k celkově nízkému počtu dostupných instrukcí. Druhým důvodem je použití ostatních instrukcí pouze ve fázi inicializace/terminace obslužné aplikace pro nastavení a zpětné čtení některých parametrů. Ukončení programu je doporučenou pomocí instrukce ECHO, která se vyšle jako poslední. Po přijetí odpovědi od instrukce ECHO máme jistotu, že instrukční fronta je prázdná a program motoru byl vykonán kompletní.

4.2.7.5 Konceptuální model konečného automatu pro vykonávání instrukcí

Při navrhování konečného stavového automatu (obrázek 4.17) jsem se inspiroval vykonávání programu u dnešních procesorových jader. Vykonání jedné instrukce prochází celkem 4–mi fázemi:

- IDLE – klidový stav, instrukční fronta je prázdná
- LOAD – načítání instrukce z instrukční fronty
- DECODE – na základě operačního kódu instrukce přejdu do stavu příslušného dané instrukci (instrukcím)
- EXECUTE – zde dojde k provedení instrukce – počet dílčích stavů pro vykonání celé instrukce se může lišit

Po vykonání fáze EXECUTE se provede návrat do stavu IDLE, kde se při neprázdnosti instrukční fronty začne vykonávat další smyčka provádění instrukce. Zvláštní instrukce s prioritou jsou vykonávány přednostně. Všechny prioritní instrukce zároveň generují odezvu systému vysílanou přes sériovou linku.



Obrázek 4.17: Koncept FSM pro vykonávání instrukcí řídícího systému

4.2.7.6 Detailní specifikace FSM pro vykonávání instrukcí řídícího systému

Na základě konceptuálního modelu, popsaného v kapitole 4.2.7.5, jsem vytvořil konečný stavový automat (FSM) pro provádění instrukcí. Tento automat má celkem 48 různých stavů, kde některé z nich jsou sdíleny pro více instrukcí. FSM má celkem 108 přechodových hran, 21 vstupů a 37 výstupů pro řízení hardwarové části řídícího systému.

4.2.7.7 Úrovně priorit implementovaného FSM pro provádění instrukcí

V systému jsou realizovány instrukce bez priority (vykonávané in–order) a instrukce s vyšší prioritou (vykonávány out–of–order). Kromě nich jsou v systému další úrovně priorit přiřazené událostem, které nejsou vyvolány instrukcemi, ale v daném systému mohou nastat. Typické události v systému mohou být: aktivace sledovacího obvodu (watchdog) nebo je motor mimo povolenou oblast.

Priority použité v systému:

1. Watchdog
2. Zahlcení některé z fronty
3. Odpočítáno 256 instrukcí MOVE/WAIT
4. Obsluha prioritní instrukce v tomto pořadí:
   (a) RESETCMD
   (b) FLUSH
   (c) HALT
   (d) RESUME
5. Motor je mimo dovolenou oblast
6. Přepnutí stavu magnetické brzdy
7. Obsluha alarmu od řídící jednotky ESMC–C2
8. Normální instrukce

Jak je vidět na úrovních priorit, události jsou obsluhovány od nejdůležitějších k nejméně důležitým. Sledovací obvod (Watchdog) detekuje neprázdnost instrukční fronty, kdy po dobu delší než 10 minut nebyla detekována žádná změna na budících výstupech motoru. Tato konfigurace parametrů má tedy detekovat zablokování hardwarové části řídícího systému.

Na pořadí obsluhy prioritních instrukcí je vidět pokus o zvýšení „pasivní“ bezpečnosti zařízení. Všechny „zastavovací“ události mají vyšší prioritu než události pro obnovení činnosti. To by mělo zajistit bezpečnou funkci zařízení za každých podmínek.

4.2.7.8 Časování standardních instrukcí

Jak bylo zmíněno v kapitole 4.2.7.5, každá instrukce prochází při svém zpracováním smyčkou IDLE – LOAD – DECODE – EXECUTE. Fáze EXECUTE je pro každou instrukci odlišná. Znalost přesného časování každého děje v řídícím systému je důležitá pro možnost predikovat chování hardwarové části řídícího systému. Všechny časy provádění jednotlivých instrukcí jsou udávány diskrétním čase – násobku hodinových taktů nutných pro vykonání.

Instrukce MOVE

Provádění instrukce začíná stavem S_MOVE, který spouští funkční blok (kapitola 4.2.5) pro generování buzení motoru. Po něm následuje stav S_MOVE_RUNNING, který se kryje s činností funkčního bloku. Stav S_MOVE_DONE čeká na ukončení činnosti funkčního bloku. Výraz pro výpočet času nutného pro vykonání instrukce:

$$Time = (2 \times Prescaler + 1) \times Distance + 6 + 2 + 3\ [Cycles]$$

Instrukce WAIT

Instrukce WAIT využívá stejný mechanismus jako instrukce MOVE (kapitola 4.2.5). Parametr **Distance** je pro tuto instrukci nastaven na konstantní hodnotu 1. Výsledný výraz pro instrukci WAIT:

$$Time = (2 \times Timer + 1) + 6 + 2 + 3 \ [Cycles]$$

Instrukce SET_POS, SET_LIMIT_1, DISEN_LIMIT_1, EN_LIMIT_1, SET_LIMIT_2, DISEN_LIMIT_2, EN_LIMIT_2, SET_TIME

Tyto instrukce využívají stejný mechanismus obsluhy. Výše jmenované instrukce negenerují odezvu vysílanou přes sériovou linku – UART. Počet taktů nutných pro tyto instrukce je 4.

Instrukce ECHO, RELAY1_DOWN, RELAY1_UP, RELAY2_DOWN, RELAY2_UP, BRAKE_EN, BRAKE_DIS, ALARM_CLEAR

Tyto instrukce generují odezvu vysílanou přes sériovou linku. Délka vysílané odezvy je jeden byte. Počet taktů nutných pro tyto instrukce je 5.

Instrukce READ_POS, READ_R_POS, READ_TIME, STATUS_REQ

Instrukce se složitější odezvou typicky obsahují informaci obsaženou ve stavovém registru řídící jednotky. S výjimkou instrukce STATUS_REQ tyto instrukce vrací datový typ Integer (32–bitový). Odezva je zapsána naráz do výstupního 5–bytového vyrovnávacího registru. Poté je pomocí stavů SHIFT1 až SHIFT5 vysouvány byty do zápisového portu fronty pro vysílání odezev prostřednictvím sériové linky. Počet taktů nutných pro tyto instrukce je 9 (STATUS_REQ má délku 8).

4.2.7.9 Časování prioritních instrukcí RESUME, HALT, FLUSH

Prioritní instrukce se vykonávají mimo pořadí. Z tohoto důvodu je jejich časování složitější, protože je nutné odvodit i čas, kdy dojde k aktivaci požadované funkčnosti. Prioritní instrukce se mohou aktivovat v průběhu stavu S_MOVE_RUNNING, který odpovídá běhu modulu pro generování buzení lineárního motoru, případně příkazu WAIT. Ostatní instrukce vzhledem ke své délce nemá smysl přerušovat uprostřed běhu.

Výraz pro výpočet času aktivace funkčnosti vychází z doby nejdelší atomické instrukce (9 taktů) a doby nutné pro obsluhu prioritní instrukce (4 takty). V případě nejhoršího případu u neatomické instrukce (MOVE/WAIT) je nutná doba pro dosažení stavu S_MOVE_RUNNING (5 taktů) plus doba obsluhy (4 takty). Z těchto výpočtů je patrné, že prioritní instrukce se provede do 9 taktů od jejího přijmutí modulem pro příjem přes sériovou linku – UART.

Tyto prioritní instrukce generují odezvu o velikosti jednoho bytu z důvodu své důležitosti na řízení systému.

4.2.7.10 Časování prioritní instrukce RESET_CMD

Tato prioritní instrukce nemá žádné relevantní časování, neboť po jejím dekódování je celá řídící jednotka v následujícím taktu uvedena do stavu po zapnutí (Power–On state).

4.2.7.11 Časování událostí vznikajících uvnitř řídící jednotky

Časování událostí vznikajících uvnitř řídící jednotky se obsluhují stejným mechanismem jako obsluha prioritních instrukcí. Seznam událostí vznikajících v řídící jednotce:

- Aktivace watchdogu – násilně ukončí instrukci MOVE/WAIT v modulu pro jejich zpracování a umožní návrat obslužného stavového automatu do stavu IDLE. O proběhnutí této procedury je nadřízená vrstva řídícího systému informována prostřednictvím odezvy vysílanou přes sériovou linku.

- Zahlcení výstupní(ch) front(y) – u řídící jednotky se předpokládá dřívější zahlcení instrukční fronty před zahlcením fronty pro vysílání odezev přes sériovou linku. To je dáno nepoměrem mezi počtem možných záznamů, které lze do jednotlivých front uložit. Mělo by tedy být možné při zahlcení fronty příkazů stále vložit odezvu s informací o zahlcení do fronty určené pro vysílání odezev.

  Tato událost by měla být zároveň přístupná obsluze nevymaskovatelným způsobem, například přivedením výstupu na LED signalizující nestandardní událost.

- Provedeno 256 instrukcí MOVE/WAIT – Tato událost je vyžadována pro správnou činnost komunikačního protokolu pro komunikaci mezi hardwarovou a softwarovou částí řídícího systému. Komunikační protokol je popsán v kapitole 4.2.7.4.

- Motor opustil povolenou oblast – Signály L1_REACHED a L2_REACHED nabývají hodnoty logické jedničky v okamžiku rovnosti příslušného registru limity a registru polohy. O proběhnutí této procedury je nadřízená vrstva řídícího systému informována prostřednictvím odezvy vysílanou přes sériovou linku.

- Manuální aktivace/deaktivace magnetické brzdy – Řídící jednotka umožňuje obsluze manuálně ovládat stav magnetické brzdy. Protože řídící signál pochází od tlačítka na desce plošných spojů, je nutné předřadit zákmitový a metastabilní filtr. Protože zákmitový filtr pracuje v oblasti milisekund, nemá smysl řešit přesný (v jednotkách taktů) časový výraz pro tuto funkci řídící jednotky.

- Alarm řídící jednotky ESMC–C2 – Tento signál indikuje nemožnost provést požadovaný příkaz. Typickou příčinou je pokus programu pokračovat v činnosti motoru mimo jeho povolené provozní parametry (příliš vysoká rychlost, snaha o pohyb mimo mechanické meze).

  V takovém případě je nutné zastavit činnost řídící jednotky a informovat nadřízenou vrstvu o této události. Doporučený postup obsluhy ve vyšší vrstvě je vyslání instrukce FLUSH a ALARM_CLEAR. Instrukce ALARM_CLEAR vygeneruje signál pro odblokování řídící jednotky ESMC–C2.

4.2.7.12 Kódy a formát odezvy pro instrukce s odezvou a vnitřní signály řídící jednotky

Pokud je součástí odezvy i hodnota (velikost odezvy je 5 bytů), je první byte hlavička a následující čtyři byty jsou reprezentací 32–bitového čísla (Integer). Výjimkou je odpověď na instrukci STATUS_REQ, která odpověď o více částech a její celková délka je 4 byty. Odezvy generované instrukcemi jsou v tabulce 4.5.

Odezvy generované událostmi řídícího systému jsou v tabulce 4.7. Odezvy na události od synchronizačních vstupů jsou shodné s jejich ekvivalentní funkcí mezi instrukcemi (RESUME, FLUSH). Odezva na manuální ovládání magnetické brzdy závisí na předchozím stavu brzdy a podle nového stavu brzdy je vyslána stejná odezva jako v případě příslušné instrukce, které jsou v tabulce 4.5.

Odezvy instrukcí generované řídící jednotkou			
Instrukce generující odezvu	Binární reprezentace odezvy – hlavička	Délka odezvy	Význam předávané hodnoty
HALT	1101 0001	1	Není
FLUSH	1101 0011	1	Není
RESUME	1101 0010	1	Není
READ_POS	0100 0101	5	Pozice motoru
READ_R_POS	0001 0101	5	Pozice získaná z kvadraturního enkodéru
READ_TIME	0010 0101	5	Času jednotky
BRAKE_EN	11110110	1	Není
BRAKE_DIS	11110111	1	Není
STATUS_REQ	1010 0100	4	Stavové informace, struktura odpovědi – tabulka 4.6
ECHO	1100 0011	1	Není
RELAY1_DOWN	1110 0000	1	Není
RELAY1_UP	1110 0001	1	Není
RELAY2_DOWN	1110 0010	1	Není
RELAY2_UP	1110 0011	1	Není
ALARM_CLEAR	0101 1010	1	Není

Tabulka 4.5: Odezvy instrukcí generované řídící jednotkou

Struktura odezvy na instrukci STATUS_REQ				
Bit(y)	39 – 32	31 – 27	26	25
Význam	Hlavička odpovědi	0000	Magnetická brzda	Limita 1
Bit(y)	24	23 – 19	18 – 8	7 – 0
Význam	Limita 2	00000	Počet instrukcí v instrukční frontě	0000 0000

Tabulka 4.6: Struktura odezvy na instrukci STATUS_REQ

Odezvy generované událostí řídící jednotky		
Událost generující odezvu	Binární reprezentace odezvy – hlavička	Délka odezvy
WATCHDOG	1101 0001	1
QUEVE_OVERLOADED	1101 0011	1
NEXT_256	1101 0010	1
REACHED_L1	0100 0101	1
REACHED_L2	0001 0101	1
BRAKE_MANUAL	0010 0101	1
ALARM	11110110	1

Tabulka 4.7: Odezvy událostí generované řídící jednotkou

4.2.8 Podpůrné obvody v hardwarové části řídícího systému

Pro správnou funkci řídící jednotky jsou nutné i další obvody, které mají však jen podpůrnou funkci a do samotné funkce řídícího systému nezasahují zásadním způsobem.

4.2.8.1 Zákmitový filtr

Zákmitový filtr slouží k odfiltrování zákmitů na signálech, jejichž hodnota závisí na mechanickém stavu součástky, například tlačítka. Tlačítko může při přepínání kontaktů generovat falešné pulsy na vstupním signálu. Z tohoto důvodu je nutné po první změně signálu počkat určitou dobu na ustálení hodnoty a zkontrolovat, zda se hodnota signálu skutečně změnila.

Realizovaný zákmitový filtr má generický parametr COUNTER_LENGHT má výchozí hodnotu 5000 taktů, což při taktovacím kmitočtu 50 MHz dává čas 1 ms na ustálení signálu. Součástí zákmitového filtru je i metastabilní filtr .

4.2.8.2 Synchronizační detektory

Jedním z požadavků na funkčnost řídící jednotky je i možnost spolupráce s ostatními přístroji používaných v laboratoři. Z tohoto důvodu je nutné přidat obvod, který bude reagovat na synchronizační signály. V této řídící jednotce se budou vyskytovat 2 typy synchronizačních signálů: signál pro zahájení činnosti řídící jednotky (externí varianta příkazu RESUME) a signál pro bezpečné zastavení činnosti řídící jednotky (externí varianta příkazu FLUSH).

VHDL soubor s detektorem umožňuje prostřednictvím generických parametrů nastavit počet bitů detektoru (pro možnost tvorby více synchronizačních vstupů) a zda detektor má detekovat sestupnou nebo vzestupnou hranu signálu.

4.2.9 Závěrečné shrnutí části realizované v FPGA

Výsledná část řídícího systému realizovaná v FPGA je poměrně komplexním systémem, který naplňuje myšlenku systému na čipu (System on a Chip – SoC). Takový systém se vyznačuje integrací procesorového jádra (hlavní modul pro zpracování instrukcí) pro zpracování instrukcí programu, různé druhy pamětí (fronty). V takovém systému jsou dále i funkční bloky (PWM pro generování budících signálu pro řídící jednotku ESMC–C2) a periferní obvody pro komunikace s vnějším světem (sériová linka – UART).

Systém řídící jednotky podporuje 27 různých instrukcí, 5 různých vnitřních událostí a 3 různé vnější události. Vzhledem k realizaci určité „pasivní bezpečnosti“ a generování odezvy pro každou důležitou událost lze tento systém považovat za reaktivní.

4.3 Elektrické zapojení a tvorba desky plošných spojů

Nejnižší úrovní řídícího systému je elektrické zapojení všech částí systému. Jak bylo zmíněno v kapitole 4.1.4, během fáze realizace vznikla celkem tři různá elektrická zapojení pro propojení čipu FPGA a řídící jednotky lineárního motoru ESMC–C2.

V průběhu realizace bylo nutné vytvořit několik desek plošných spojů obsahující doplňující elektronické součástky (optočleny, napěťové regulátory, LED, tlačítka), které sloužily k ověření funkčnosti některých dílčích návrhů. Na základě měření a testování byly provedeny úpravy zapojení, na základě kterých se připravovaly nové prototypy. Během těchto následných realizací bylo nutné změnit použitý čip FPGA.

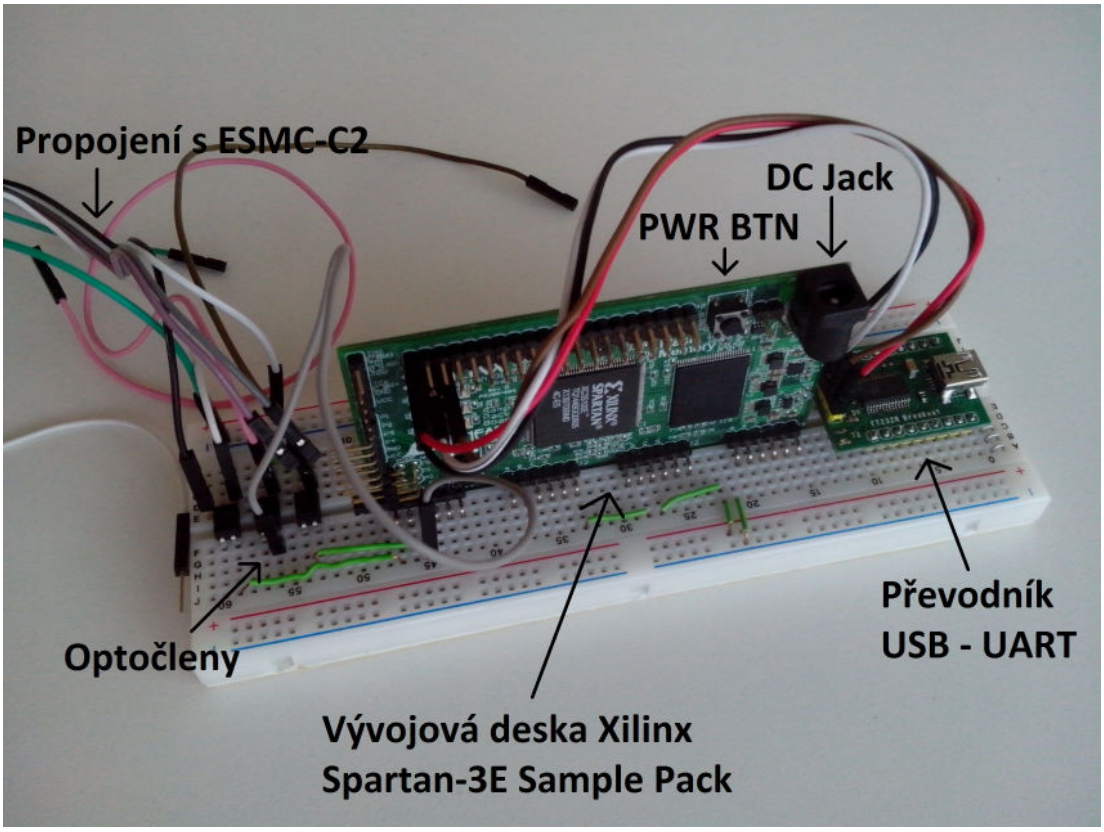
4.3.1 Verze 0 – prvotní prototyp zapojení realizovaný na nepájivém poli

Prvotní seznámení a odzkoušení funkcionality řídící jednotky ESMC–C2 a jejím režimem Driver (kapitola 4.1) jsem řešil improvizovaným zapojením vývojové desky Xilinx Spartan–3E Sample Pack (kapitola 4.2.1). Při osazení pinovými lištami s roztečí 2,54 mm je možné tuto vývojovou desku zasunout do nepájivého kontaktního pole. Takovéto zapojení je znázorněno na obrázku 4.18.

Na nepájivém poli jsem úspěšně otestoval první verzi zapojení, na kterém jsem pracoval se signály pro buzení motoru, magnetickou brzdu a kvadraturní enkodér řídící jednotky ESMC–C2. Tím jsem zjistil správné zapojení jednotlivých pinů na konektoru pro komunikaci s řídící jednotky ESMC–C2, správné signálové průběhy, správné napěťové úrovně a napájecí napětí pro všechny části zapojení.

V této fázi jsem udělal chybu týkající se galvanického zapojení. Pro možnost sledování signálových průběhů na osciloskopu jsem spojil země obou původně izolovaných částí. Tuto chybu jsem odstranil až v poslední verzi desky plošných spojů (DSP).

Na této verzi zapojení jsem implementoval na FPGA první demonstrační aplikaci týkající se kvadraturního enkodéru. Po této demonstrační aplikaci pro účely ověřování a měření jsem začal z vývojem systému zpracovávající navržený instrukční soubor. Postupem času se začaly projevovat nevýhody tohoto zapojení (realizace stylem „vrabčí hnízdo“). Z nepájivého pole se často uvolňovaly propojovací vodiče, které způsobovaly nesprávnou činnost zapojení. Hledání těchto poruch zabralo mnoho času, který se dal strávit realizací a testováním nové řídící jednotky.



Obrázek 4.18: Zapojení všech důležitých částí ve verzi 0 – ilustrační zapojení

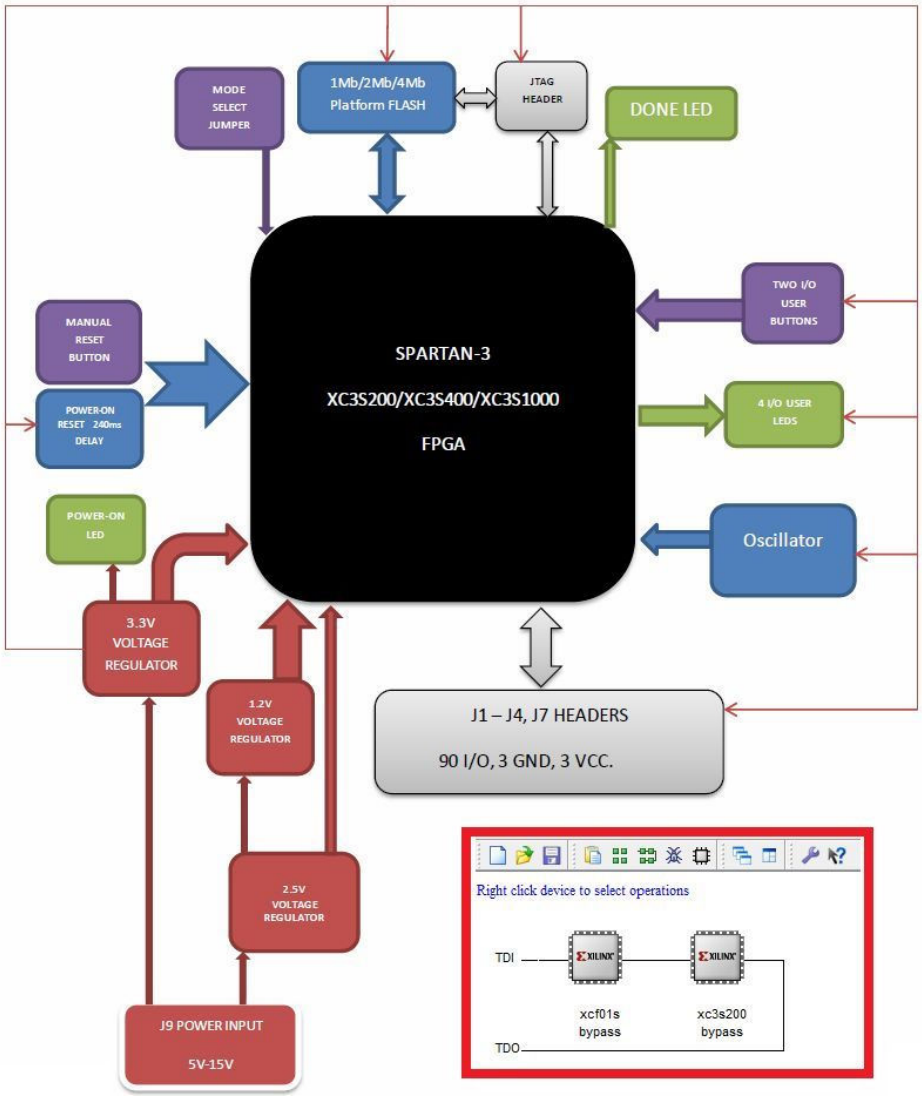
Dalším problémem se ukázala použitá vývojová deska Xilinx Spartan–3E Sample Pack. Tuto vývojovou desku nebylo možné pevně připevnit k případné rozšiřující desce díky absenci montážních otvorů na DPS. Použitý konektor pro napájení (DC Jack) nemá žádnou pojistku proti samovolnému odpojení napájecího kabelu. Tím bych zapojení přidal další potencionálně nespolehlivý prvek. Dále je nutné při každém připojení napájecího napětí stisknout tlačítko PWR, které zapne napájecí kaskádu na vývojové desce.

Druhým problémem s vývojovou deskou bylo nahrání bitstreamu FPGA do paralelní paměti typu Flash. Firma Xilinx poskytovala pro tuto desku nástroj pro nahrání bitstreamu, který byl založen na modifikované verzi procesorového jádra MicroBlaze. Tato aplikace byla dodávána ve formě projektu se zdrojovými kódy pro Xilinx ISE ve verzi 4.3. Tato verze je velmi dlouho nepodporovaná a i přes usilovné snažení se nepodařilo tuto aplikaci přeložit. Před každým použitím desky bylo nutné FPGA naprogramovat ručně prostřednictvím hardwarového programátoru (Xilinx Platform Cable).

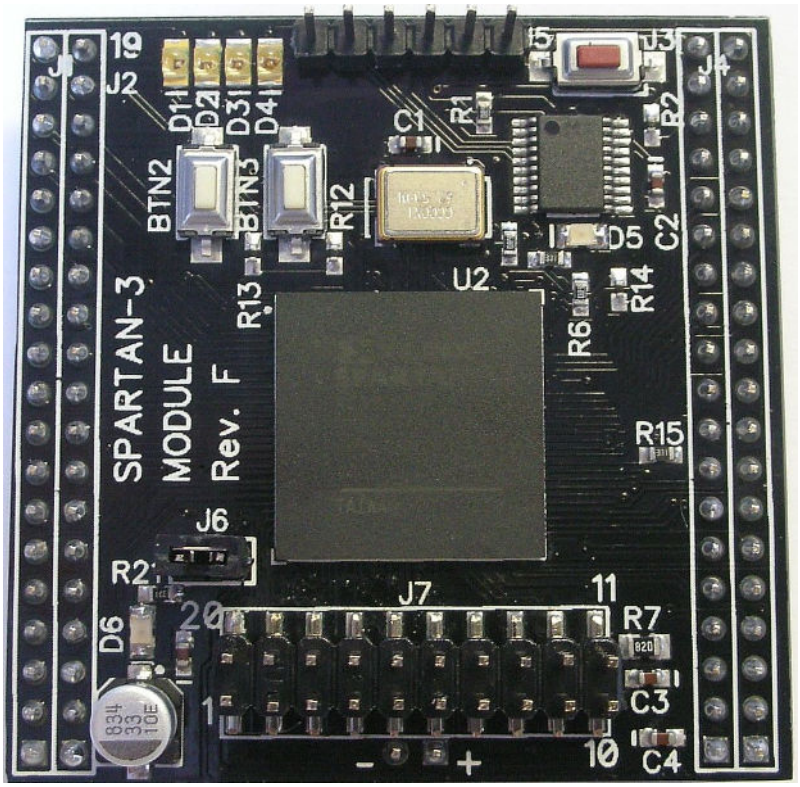
Na základě výše zmíněných skutečnosti jsem se rozhodl vytvořit nové elektrické zapojení, které bude realizováno na nově vytvořené desce plošných spojů a nové vývojové desky s FPGA, pokud možno v průmyslovém provedení. Nově použité FPGA by se mělo co nejvíce blížit parametrům čipu Xilinx Spartan–3E 100K.

4.3.2 Verze 1 – prototyp s průmyslovou vývojovou deskou a vlastním návrhem desky plošných spojů

Z důvodů zmíněných u zapojení ve verzi 0 jsem se rozhodl pro nákup nové vývojové desky s čipem FPGA co nejvíce podobném čipu na desce Xilinx Spartan–3E Sample Pack. Na základě hledání vhodné desky jsem se rozhodl pro desku estonské výroby firmy FPGA.STORE. Tato firma nabízí své vývojové desky především prostřednictvím mezinárodního aukčního portálu eBay. Deska byla zvolena na základě nízké ceny, dostupnosti dobré dokumentace a funkčností. Možnosti desky jsou shrnuty na obrázku 4.19, reálný vzhled je na obrázku 4.20.



Obrázek 4.19: Možnosti nové vývojové desky s čipem Xilinx Spartan–3 200K



Obrázek 4.20: Reálný vzhled nové vývojové desky s čipem Xilinx Spartan–3 200K

4.3.2.1 Možnosti nové vývojové desky

Nová vývojová deska s čipem Xilinx Spartan–3 200K obsahuje:

- FPGA Xilinx Spartan–3 200K (XC3S200–FT256–4C)
- ROM pro uložení bitstreamu XCF01S
- Oscilátor 50 MHz
- 90 I/O portů
- 4×LED
- 2×tlačítko
- Lineární regulátory pro napájení desky a FPGA
- Obvod pro zajištění resetu při náběhu napájení

Vývojová deska používá standardní dvouřadé pinové lišty pro všechny I/O piny i pro napájení desky. Při návrhu DPS s potřebnými protikusy dosáhneme spolehlivého spojení mezi vývojovou deskou a DPS. Tím odpadnou problémy se špatnými kontakty.

4.3.2.2 Rozbor čipu Spartan–3 200K

Na základě tabulek 4.8 a 4.1 můžeme porovnat klíčové parametry obou čipů FPGA. Nový čip Xilinx Spartan–3 200K [9] má 2× více bloků pro realizaci logických výrazů a až 3× více předpřipravených bloků (BlockRAM, DCM).

Na základě těchto zjištění je možné změnit cílový čip a přenést celý projekt bez úprav ve zdrojových kódech systému. Dále můžeme tedy zvětšit datové fronty pro frontu instrukcí a frontu pro vysílání přes sériovou linku – UART. Jedinou nutností bylo změnit namapování signálů na piny nového čipu.

Dostupné zdroje v čipu Xilinx XC3S200–FT256–4C	
Maximální počet I/O portů	173
Počet Sliců	1920
Počet LUTů	4320
Počet klopných obvodů	4320
Velikost BlockRAM	216 kbit
Počet BlockRAM	12
Digital Clock Managment (DCM)	4
Dedikované násobičky 18x18 bitů	12
Rychlost obvodu	-4 (standardní)
Typ obvodu	pro běžné použití

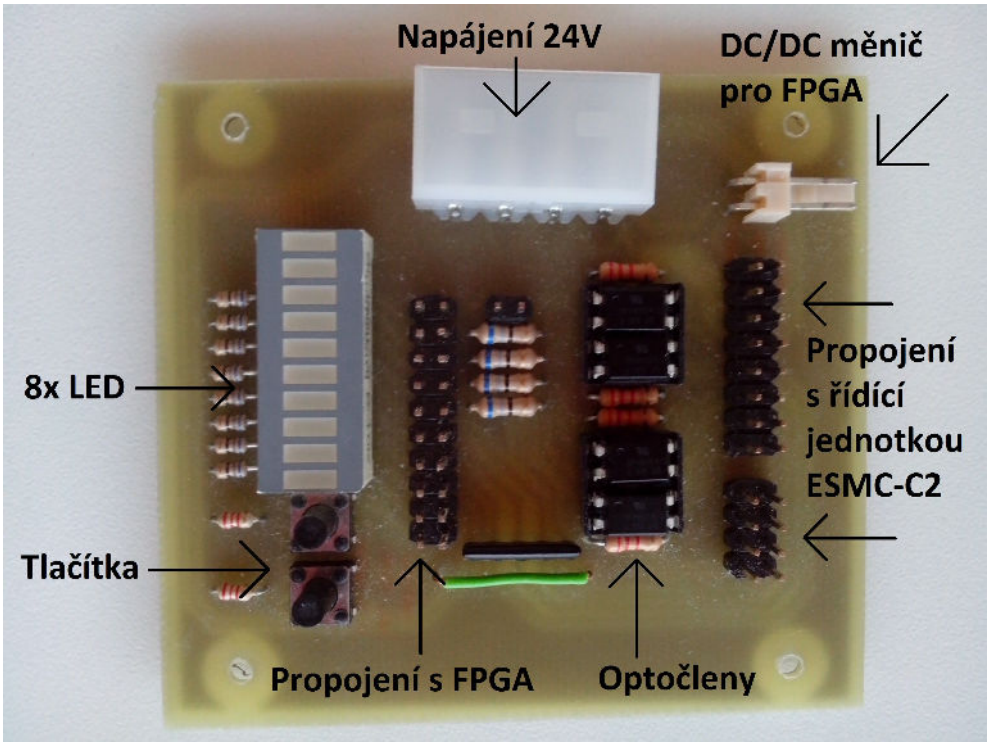
Tabulka 4.8: Dostupné zdroje v čipu Xilinx XC3S200–FT256–4C

4.3.2.3 Návrh a realizace desky plošných spojů pro vývojovou desku s čipem Xilinx Spartan–3

Rozhodl jsem se zapojení použité ve verzi 0 překreslit v návrhové systému Eagle a vyrobit desku plošných spojů. Tím bych měl zvýšit spolehlivost celého zapojení. Zvýšení spolehlivosti umožní testování a měření provozních parametrů v delším časovém horizontu bez rizika nechtěného rozpojení některého vodiče. Schéma (obrázek D.2) celého zapojení a výsledná podoba desky (obrázek D.1) jsou v příloze D. K propojení s novou vývojovou deskou s čipem Spartan–3 200K slouží 20-pinová dvouřadá lišta, která se prostřednictvím kabelu napojuje na 20-pinovou lištu na vývojové desce (konektor J7).

Napájení desky plošných spojů se napájí standardním MOLEX konektorem používaným v počítačích. Napájení FPGA se řeší prostřednictvím DC/DC měniče, kdy dochází ke konverzi z napětí +24 V na +5 V použitelných pro napájecí kaskádu FPGA.

Obrázek 4.21 znázorňuje jednotlivé funkce realizované desky plošných spojů. Obrázek 4.22 znázorňuje propojení všech použitých částí v elektrickém zapojení.



Obrázek 4.21: Realizovaná deska plošných spojů pro první verzi elektrického zapojení

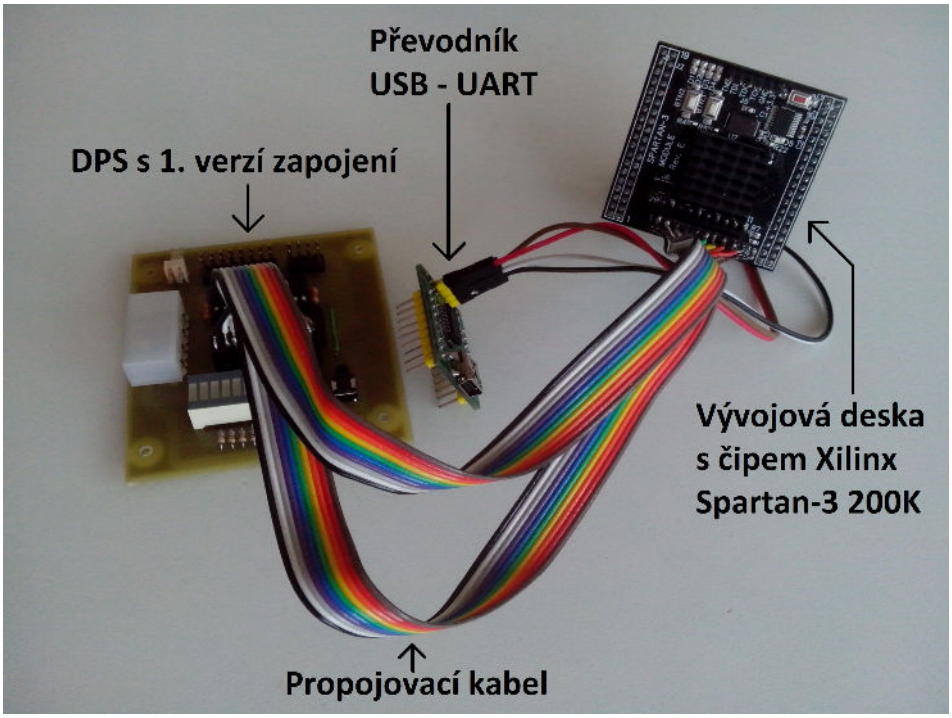
4.3.2.4 Závěry získané z realizace první verze desky plošných spojů

Na základě testů při „plném provozu“ bylo učiněno několik zjištění. Všechny navržené funkce fungovaly korektně. Problémem se stalo zapojení optočlenů, které v klidovém stavu zůstávaly trvale sepnuté. Teplota na optočlenech se po hodině v klidovém stavu pohybovala nad 60 °C. Optočleny mohou snášet teploty až do +125 °C, ale pro dlouhodobý provoz je dobré držet jejich teplotu řádově na teplotě okolního prostředí. Vývojová deska s čipem Spartan–3 byla shledána jako vyhovující. Hlavní změnou vůči předchozí vývojové desce je použití ROM čipu pro uložení bitstreamu FPGA, který lze snadno naprogramovat z vývojového prostředí Xilinx ISE nástrojem iMPACT. Není tedy nadále nutné před každým použitím FPGA ručně nahrát celý systém.

4.3.3 Verze 2 – prototyp s průmyslovou vývojovou deskou a vlastním návrhem desky plošných spojů – pokus o finální DPS

Po ověření funkčnosti zapojení první verze a některých dalších částí systému jsem provedl další iteraci v návrhu a realizaci řídícího systému. Cílem této iterace mělo být zlepšení provozních parametrů systému a rozšíření o další funkcionalitu, případně integrace již otestované funkcionality na jednu desku plošných spojů.

Předchozí verze desky plošných spojů obsahovala 4 kanály pro výstupy a 4 kanály pro vstupy s konverzí napěťových úrovní +24 V na + 3,3 V. Napěťová úroveň +3,3 V odpovídá



Obrázek 4.22: Realizovaná deska plošných spojů pro první verzi elektrického zapojení – ostatní části zapojení (DC/DC měnič pro napájení FPGA není vyfocen)

maximálnímu dovolenému vstupnímu napětí pro použitý čip FPGA Spartan–3. Řídící jednotka ESMC–C2 umožňuje komunikaci po 8 vstupech a 8 výstupech, proto další verze desky by měla disponovat 8–mi výstupními a 8–mi vstupními kanály. Desku tak nebude nutné předělávat v případě doplňování další funkčnosti řídícího systému.

4.3.3.1 Převodník z rozhraní USB na sériovou linku – UART

Druhou součástí mělo být integrace čipu pro převod rozhraní USB na sériovou linku – UART. V dnešní době jsou dostupné 3 hlavní typy těchto převodníků. Jmenovitě jde o čipy:

- Prolific PL2303 – Dostupný v pouzdře SOIC, nepodporován na Windows 8/8.1.
- Silicon Labs CP2102 – Dostupný v pouzdře QFN, podporován všemi OS.
- FTDI FT232RL – Dostupný v pouzdře SOIC, případně dalších, podpora všemi OS.

Z důvodů dopředné kompatibility jsem vyřadil obvod PL2303. Pro snadnou realizaci desky plošných spojů v systému Eagle je jednodušší použít součástku s pouzdrem typu SOIC, proto jsem se rozhodl pro použití obvodu FTDI FT232RL. Při navrhování elektrického zapojení jsem vycházel z katalogového zapojení doporučovaného výrobcem. Elektrické zapojení, které jsem použil, je na obrázku E.1. Toto zapojení obsahuje propojovací header, který pro účely testování nebo poruchy umožňuje snadno připojit jiný převodník na rozhraní UART.

4.3.3.2 Napájení DPS a vývojové desky s FPGA

Třetí částí nově vyvíjené desky plošných spojů je osadit desku DC/DC měničem. Tím by bylo možné použít na desce pouze jeden napájecí konektor s jedním napájecím napětím pro napájení celé řídící jednotky. Protože hotové DC/DC měniče ve formě modulů je obtížné sehnat v České Republice za přijatelnou cenu, hledal jsem vhodný modul u prodejců na dálném východě.

Dalším požadavkem na DC/DC měnič bylo přijatelné mechanické rozměry a způsob připojení k desce plošných spojů. Na základě těchto požadavků jsem se rozhodl pro miniaturní modul DC/DC měniče (obrázek 4.23), který je možný k desce připojit prostřednictvím standardních pinových lišt s roztečí 2,54 mm. Použitý DC/DC modul využívá čip MP1584. Parametry tohoto obvodu jsou v tabulce 4.9.

Obrázek 4.23: Miniaturní modul DC/DC měniče s čipem MP1584

Parametry čipu MP1584 – DC/DC měniče	
Parametr	Hodnota
Vstupní napětí	4,5 V – 28 V
Výstupní napětí	1,3 V – 17 V
Výstupní proud	3 A (2 A bez chladiče)
Pracovní frekvence	1,5 MHz
Účinnost	95 %
Provozní teplota	−20 °C — +85 °C
Rozměry	22 mm × 17 mm

Tabulka 4.9: Parametry čipu MP1584 – DC/DC měniče

4.3.3.3 Dodatečné ústní požadavky na funkcionalitu zapojení desky plošných spojů

Poslední částí bylo nutné zajistit připojení vývojové desky s čipem Spartan–3. Tato vývojová deska má vyvedeny I/O piny na standardní dvouřadé pinové lišty 2x19 pinů, na každé straně desky po jednom bloku. Pro propojení s navrhovanou deskou plošných spojů stačí použít protikusy pinových lišt, které zajistí i určitou mechanickou odolnost.

Protože tato verze desky plošných spojů mohla být označena za finální, byl ze strany pracovníků Fakulty strojní vznesen ústní požadavek na snadnou opravitelnost celé řídící jednotky. Tento požadavek byl řešen umístěním všech dílčích připojovaných částí do patic na desce plošných spojů. Výjimkou jsou pouze některé základní diskrétní součástky a konektory.

4.3.4 Elektrické zapojení desky plošných spojů pro verzi 2

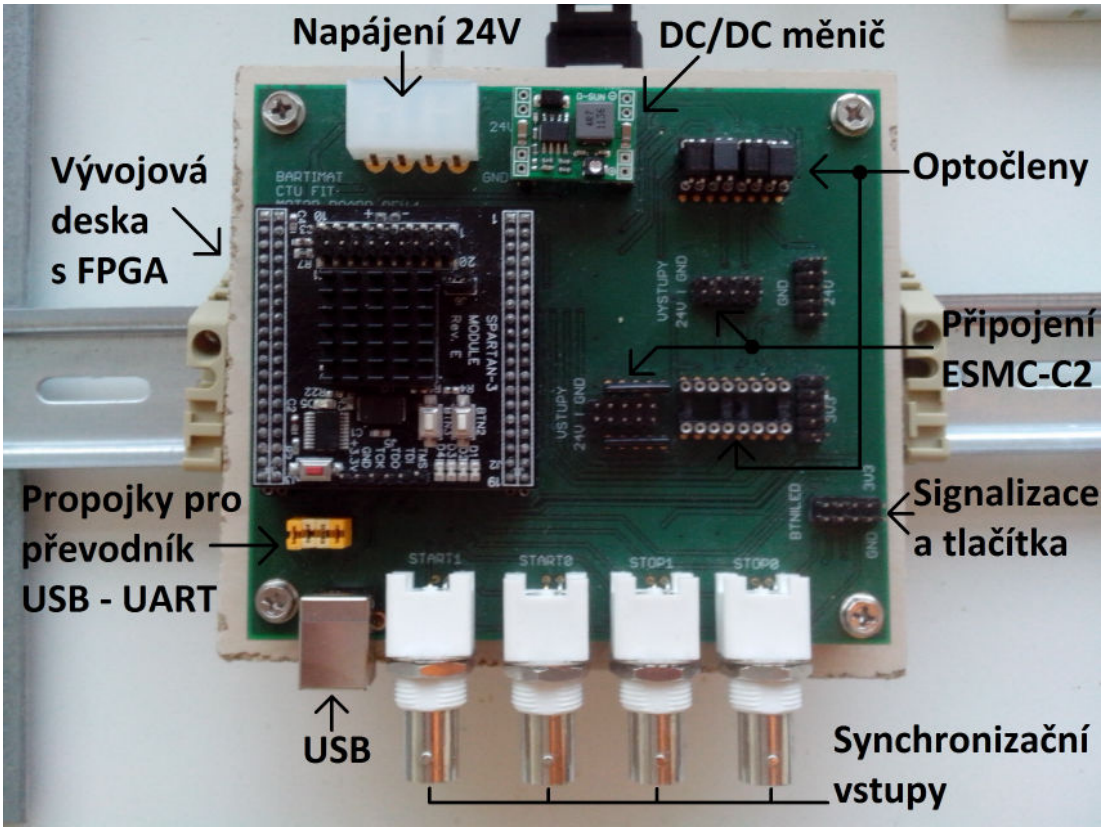
Schéma využívá dílčího zapojení popsané v kapitole 4.1.4.2 (druhá verze zapojení mezi FPGA a řídící jednotkou ESMC–C2). Ve své podstatě jde pouze o propojení pinových lišt s 8–mi násobným zapojením 4.1.4.2. Výraznější změnou je přidání obvodu 74LVX245 pro zajištění odolnosti synchronizačních vstupů při připojení signálů s napěťovou úrovní +5 V. Celé zapojení desky plošných spojů v druhé verzi je v příloze E.

Výsledný vzhled desky plošných spojů s osazenou vývojovou deskou s FPGA, DC/DC měničem a optočleny je na obrázku 4.24. Každá tato část je umístěna v patici a je tedy možné poškozenou část rychle a snadno vyměnit.

4.3.4.1 Poznatky získané z realizace druhé verze zapojení

Na desce plošných spojů ve verzi 2 jsem provedl testy další požadované funkcionality. S tím souvisí i proces testování celého systému. Pro účely otestování hardwarové částí řídícího systému byla vytvořena aplikace pro ověření komunikačního protokolu a funkčnosti zařízení během dlouhodobých testů (popsáno detailně v kapitole 5.2).

Na základě těchto testů jsem došel k řadě zjištění. Při dlouhodobém provozu řídící systém často ukončil svojí činnost bez zjevného důvodu. Na základě 9 měsíců hledání původu této chyby jsem dospěl k závěru, že řídící systém nesplňuje požadavky na elektromagnetickou kompatibilitu. Během testování o celkové době 9 měsíců se projevily i další různé nedostatky. Dospěl jsem k závěru, že je nutné navrhnout a realizovat další verzi desky plošných spojů. Na základě nově získaných informací jsem se rozhodl pro návrh, který by měl vyhovovat požadavkům na EMC, případně by bylo možné získat certifikát ESČ pro celé elektronické zařízení od Elektrotechnického zkušebního ústavu.



Obrázek 4.24: Výsledný vzhled desky plošných spojů pro druhou verzi zapojení

4.3.5 Verze 3 – finální verze DPS s kompletně přepracovaným návrhem

Na základě poznatků v předchozí kapitole 4.3.4.1 jsem se rozhodl pro další revizi desky plošných spojů pro hardwarovou část řídícího systému [28].

Protože jsem nechtěl opakovat nezdary předchozích verzí DPS, probíhala fáze návrhu desky plošných spojů pod dohledem vyučujících z předmětu A0B34PPN. Předmět A0B34PPN má za cíl naučit studenty s návrhovým systémem Cadence OrCAD [27] pro návrh a realizaci desek plošných spojů. Dalším cílem je předat studentům znalosti pro kvalifikovaný návrh podle doporučených návrhových pravidel. V následujícím textu shrnu nedostatky druhé verze DPS a způsob jejich odstranění.

4.3.5.1 Mechanická montáž desky s hardwarovou částí řídícího systému

Všechny předchozí desky plošných spojů vznikaly bez omezení rozměrů desky a dalších mechanických záležitostí, například montážní otvory a rozmístění konektorů. Teprve následujícím krokem bylo vybrání vhodné přístrojové skříně. Rozhodl jsem pro třetí verzi tento postup otočit a rozvrhnout nejdříve umístění konektorů na vybrané přístrojové skříni. Díky tomu jsou předem dané fyzické rozměry desky plošných spojů.

4.3.5.2 Odstranění známých faktických chyb v zapojení z předchozích verzí

Použité zapojení verzí 0 – 2 pro převod signálů z napěťové úrovně +24 V na napěťové úrovně dostupné na čipu FPGA nesplňovalo požadavek na galvanické oddělení hardwarové částí řídícího systému a původní řídící jednotky ESMC–C2. Proto jsem navrhl zapojení 4.1.4.3, které využívá vysokorychlostní optočleny.

Použití vysokorychlostních optočlenů mělo dále zlepšit spolehlivost komunikace s původní řídící jednotkou ESMC–C2. Vysokorychlostní optočleny poskytují lepší tvar výsledného digitálního signálu, který je při vyšších frekvencích mnohem lépe zpracováván řídící jednotkou ESMC–C2 a nedochází tak na vyšších rychlostech k zatuhnutí řídící jednotky.

4.3.5.3 Zvýšení odolnosti vůči elektromagnetickému rušení – dodržení EMC

Z důvodu požadavků na snadnou opravitelnost celého zařízení zmíněného v kapitole 4.3.3.3 je maximum částí umístěno v paticích na desce. Tím se zvětšují plochy proudový smyček mezi jednotlivými součástkami použitých na DPS. S velikostí plochy proudové smyčky roste náchylnost k elektromagnetickému rušení. Velká proudová smyčka funguje jako anténa, která zároveň přijímá cizí rušení a vyzařuje rušení vlastní [28, 24, 21].

To je důvod, proč není dobré použít pro připojení součástek patice. Dále tyto součástky jsou vyrobeny na starších technologiích, které mají umístěny napájecí piny diagonálně na pouzdru. Pro dobrou připojitelnost veškeré kabeláže je většina konektorů na krajích desky v poměrně velké vzdálenosti od zdroje signálu. Z důvodů nízké ceny byl návrh realizován na dvouvrstvé DPS. Je tedy nutné si vystačit na signály a napájecí rozvody s pouze dvěma vrstvami mědi. To komplikuje propojitelnost a výsledkem jsou delší vodivé cesty, než by bylo nezbytně nutné. Jak je patrné, každý z těchto faktorů zvětšuje plochu proudových smyček, zvětšuje vyzařování rušivých vlivů do okolí a snižuje celkovou odolnost vůči elektromagnetickému rušení [28, 24, 21].

4.3.5.4 Zapracování dalších instrukcí od firmy Xilinx

Jak bylo řečeno v kapitole 4.3.5.3, všechny patice zvětšují celkovou plochu proudových smyček. V návrhu verze 2 je použitá průmyslová vývojová deska s čipem Xilinx Spartan–3 200K (kapitola 4.20). Hrozí zde riziko, že výrobce v brzké době ukončí produkci těchto čipů a proto z důvodu dopředné kompatibility jsem se rozhodl použít FPGA čip novější generace Xilinx Spartan–6.

4.3.5.5 Výsledné zapojení třetí verze desky plošných spojů

Výsledné zapojení je z důvodu složitosti a velikosti přiloženo v příloze F. Pro větší „profesionalitu“ DPS jsou přidány i některé přepěťové a nadproudové ochrany zařízení. Při návrhu zapojení FPGA čipu Spartan–6 jsem vycházel ze zapojení vývojové desky Papilio Pro [8].

V současném stavu je tato verze desky plošných spojů stále ve fázi návrhu – je nutné navrhnout výsledné rozložení všech součástek na DPS.

4.4 Využití zdrojů FPGA Xilinx Spartan–3 200K

Protože je v současné době provozována a testována deska plošných spojů ve verzi 2 (kapitola 4.3.3). Toto zapojení využívá vývojová desky s čipem Xilinx Spartan–3 200K (XC3S200–FT256–4C). Na obrázku 4.25 je vidět počet a typ využitých prostředků FPGA použitých pro realizaci řídícího systému. Maximální možná rychlost celého systému je 85 MHz, pro naše použití systém pracuje na 50 MHz.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	975	3,840	25%
Number of 4 input LUTs	1,250	3,840	32%
Number of occupied Slices	1,003	1,920	52%
Total Number of 4 input LUTs	1,534	3,840	39%
Number used as logic	1,244		
Number used as a route-thru	284		
Number used as Shift registers	6		
Number of bonded IOBs	45	173	26%
IOB Flip Flops	12		
Number of RAMB16s	12	12	100%
Number of BUFGMUXs	1	8	12%
Average Fanout of Non-Clock Nets	3.44		

Obrázek 4.25: Počet a typ využitých prostředků použitých pro realizaci řídícího systému

4.5 Použité hardwarové a softwarové nástroje při realizaci

Při realizaci řídícího systému bylo pro účely analýzy, návrhu a realizace nutné použít velké množství softwarových a hardwarových prostředků. Řídící systém pro FPGA byl naprogramován v jazyce VHDL ve vývojovém prostředí Xilinx ISE Design Suite ve verzi 14.3. K dílčímu ověřování funkcí systému realizovaných v FPGA jsem použil simulátor Mentor Graphic ModelSim ve verzi 10.0c.

Pro okamžité ověřování funkčnosti řídícího systému ovládaného prostřednictvím sériové linky – UART jsem používal program Eltima Advanced Serial Port Terminal [1], který mi značně zjednodušil sledování a ovládání tohoto systému.

Z hardwarových nástrojů jsem používal různé druhy osciloskopů (LeCroy WaveRunner 64Xi, Rohde & Schwarz RTO1024, GwINSTEK GDS–1152A) a logický analyzátor (Saleae Logic [14]) pro zobrazení průběhů signálů v řídícím systému [22].

Pro návrh a realizaci desek plošných spojů jsem použil systémy Eagle 6.4.0 a Cadence OrCAD 16.6.

4.6 Ověření konceptu komunikace se systémem v prostředí MATLAB

Součástí technického zadání je i požadavek na ovládání řídícího systému z prostředí MATLAB. Je nutné ověřit koncept datových struktur a komunikace se systémem v prostředí MATLAB. I bez znalosti systému MATLAB je evidentní, že je nutné provést alespoň test komunikace s hardwarovou částí systému přes sériovou linku. Na základě informací z dokumentace systému MATLAB jsem úspěšně otestoval následující kód [26, 20].

```
% Konfigurace sériové linky
s = serial('COM1','baudrate',115200,'databits',8,'parity','none','stopbits',1,
'OutputBufferSize',4096,'InputBufferSize',4096);

% Otevření portu sériové linky
fopen(s);

% Vysílané parametry
op_code = 128
speed = 1000;
distance = -100;

% Přetypování parametrů na 32-bitový Integer, v kódování Big Endian
speed_A = typecast(int32(speed), 'uint8');
distance_A = typecast(int32(distance), 'uint8');

% Vektor obsahující strukturu instrukce pro vyslání
cmd = [ob_code,speed_A(end:-1:1),distance_A(end:-1:1)]

% Vyslání instrukce přes sériovou linku
fwrite(s,cmd);

% Zavření portu sériové linky
fclose(s);
```

Čtení odezev od řídícího systému lze realizovat obdobným způsobem za pomoci volání systému MATLAB – příkaz **fread(s)**.

Kapitola 5

Testování

Testování celého řídícího systému je důležitou částí celého procesu návrhu. V tomto případě je spolehlivost řídícího systému kritická a je explicitně zmíněna v technickém zadání. Průměrná doba testování odpovídá přibližně 60% celkového času nutného na realizaci. Při návrhu řídícího systému jsem používal tyto metody testování:

- Testování simulací
- Testování demonstračním programem
- Testování elektrotechnickým vybavením

Každá z těchto metod testování se typicky používá pro různé fáze realizace. Často se však jednotlivé metody prolínají a doplňují.

5.1 Testování simulací

Testování simulací se používá při návrhu části systému, která je realizována v FPGA. Každý test se sestává z generátoru stimulů a testovaného zařízení (UUT – Unit Under Test). Pro tento typ testování jsem využíval program ModelSim. V následujících kapitolách podrobněji popíši jednotlivé testy. Soubory testů lze najít v projektovém adresáři vždy ve stejné podsložce jako testovaná část zařízení.

5.1.1 move_cmd_block_test

Tento test si klade za cíl simulaci a test funkčního bloku pro generování signálů sloužících k buzení motoru. Tento blok je popsán v kapitole 4.2.5. Test se skládá z těchto částí:

1. Test základní činnosti – vizuální test, zda se vůbec aktivuje UUT.
2. Test správné práce s aritmetikou a komplementarita příkazů – je spuštěna dvojice příkazů na stejnou vzdálenost (ale obrácenými znaménky). Kontroluje se návrat do výchozí pozice.

3. Test příkazu WAIT a ignorace druhého parametru instrukce – Každá instrukce obsahuje celkem 2 parametry. Instrukce WAIT využívá pouze jeden.
4. Test krajních a chybných hodnot příkazu WAIT – Otestovány hodnoty času 0 a 1.
5. Test krajních a chybných hodnot příkazu MOVE – Otestovány hodnoty instrukce MOVE (1,1), (0,0) a nulový posun (5,0).
6. Test dočasného zastavení modulu uprostřed probíhajícího příkazu – Důležité pro správnou funkčnost příkazu HALT.
7. Test instrukce SET_POS – Kontrola registru pozice, zda obsahuje správnou hodnotu.
8. Test násilného zastavení při výmazu programu nebo vyvolaném restartu.
9. Test znovu obnovení činnosti po instrukci HALT a FLUSH.

Na testování tohoto bloku jsem strávil velké množství času, protože tento modul nesmí způsobit zablokování celého řídícího systému. Tento test se musí provést celý i režimu simulace Post–Synthesis.

5.1.2 rs232_rx_packet_test

Tento test si klade za cíl simulaci a test funkčního bloku pro příjem instrukcí prostřednictvím sériové linky. Tento blok je popsán v kapitole 4.2.4. Test se skládá z těchto částí:

1. Test základní činnosti – příjem instrukcí SET_POS, NOP, READ_POS, SET_TIME, MOVE, WAIT.
2. Test prioritních instrukcí – Instrukce HALT, FLUSH, RESUME, RESET_CMD generují svoje vlastní výstupní signály.
3. Test dlouhé prodlevy mezi jednotlivými byty – Zjišťuji, zda může být velká časová prodleva mezi jednotlivými byty.

Test sice nepokrývá všechny realizované instrukce, ale testuje základní mechanismus rozpoznávání instrukcí, které se mohou lišit délkou celé instrukce. Zbytek realizovaných instrukcí se pokrývá testem motor_EX_top_test popsaný v kapitole 5.1.3.

5.1.3 EX_test

Tento test si klade za cíl simulaci a test bloků realizující instrukční frontu (kapitola 4.2.2.3), blok pro vykonávání instrukcí řídícího systému (kapitola 4.2.7) a test modulu pro vysílání odezev instrukcí a událostí v řídícím systému (kapitola 4.2.6).

V tomto testu je otestován celý instrukční soubor řídícího systému, předávací rozhraní z bloku pro příjem instrukcí (kapitola 4.2.4). U každé instrukce je testováno, zda vysílá správný typ odezvy. Tím je zároveň testován i blok pro vysílání odezev řídícího systému (kapitola 4.2.6). Poslední testovanou funkcionalitou je ověření komunikačního protokolu s nadřízenou vrstvou řídícího systému.

Tento test využívá služeb zjednodušeného simulačního modelu (kapitola 5.1.5) původní řídící jednotky ESMC–C2 pro simulaci některých odezev této řídící jednotky.

5.1.4 motor_driver_test_set1

Tento test si klade za cíl simulaci a test celé části řídícího systému. Vstupy a výstupy UUT odpovídají I/O pinům použitého FPGA. Veškerá komunikace se zařízením se děje prostřednictvím sériové linky – UART. Pro snadnější psaní testů jsem napsal v jazyce VHDL knihovnu **MCU_commands** s procedurami obsahující celý instrukční soubor řídícího systému.

Tento test využívá služeb zjednodušeného simulačního modelu (kapitola 5.1.5) původní řídící jednotky ESMC–C2 pro simulaci některých odezev této řídící jednotky.

5.1.5 motor_model_test

Tento test slouží pro ověření zjednodušeného simulačního modelu původní řídící jednotky ESMC–C2. Model reaguje na vstupy pro buzení motoru a magnetické brzdy. Na základě těchto vstupů generuje výstupy kvadraturního enkodéru podle dokumentace (kapitola 4.1.2).

5.1.6 Ostatní testy

Při realizaci části řídícího systému realizovaném v FPGA jsem pro účely hledání chyb a sledování průběhů signálů pomocné testy. Tyto testy jsem v soupisů testů nepopisoval pro svou jednoduchost.

5.2 Testování demonstračním programem

Další fází testování systému je ověřit jeho funkcionalitu hardwarové části řídícího systému. První testování jsem prováděl přímým zápisem a čtením hodnot, které jsem posílal prostřednictvím sériové linky do řídícího systému. K tomuto postupu jsem využíval služeb programu Eltima Advanced Serial Port Terminal (obrázek 5.1), který umožňuje sledovat data na sériové lince v několika různých formátech (ASCII, Hexadecimální, Binární).

Nevýhodou tohoto přístupu k testování je nutnost zadávat všechny instrukce ke zpracování manuálně. Pro dlouhodobější testování je tento přístup nevhodný. Z tohoto důvodu jsem si napsal obslužnou aplikaci, která bude automatizovaným způsobem dodávat patřičné instrukce.

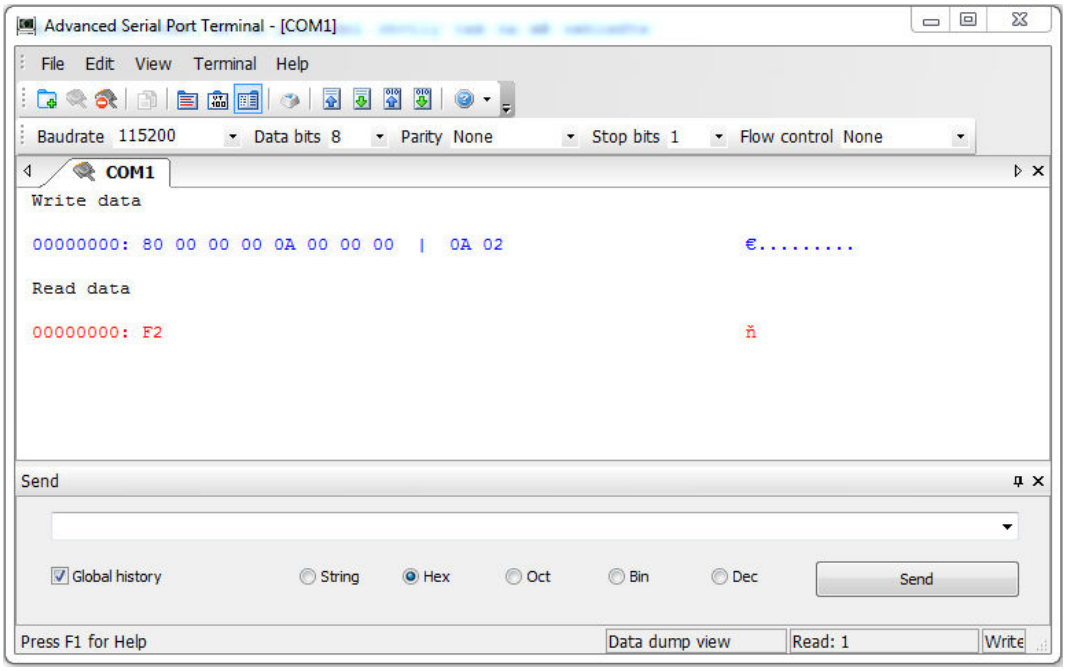
Na základě zkušeností získaných studiem jsem se rozhodl aplikaci implementovat v programovacím jazyce C, který se často používá pro tvorbu nízkoúrovňových aplikací a ovladačů.

5.2.1 Požadavky na demonstrační aplikaci

Demonstrační aplikace by měla obsluhovat novou řídící jednotku a zároveň vykonávat užitečnou činnost. Jako základní funkci jsem vybral generování „pilovitých“ průběhů. Navrženou funkcionalitu demonstruje obrázek 5.2.

Funkcionalita demonstrační aplikace:

- Generování pohybu motoru znázorněný na obrázku 5.2
- Generování jednoduchého posuvu pro nastavení výchozí pozice



Obrázek 5.1: Program Eltima Advanced Serial Port Terminal

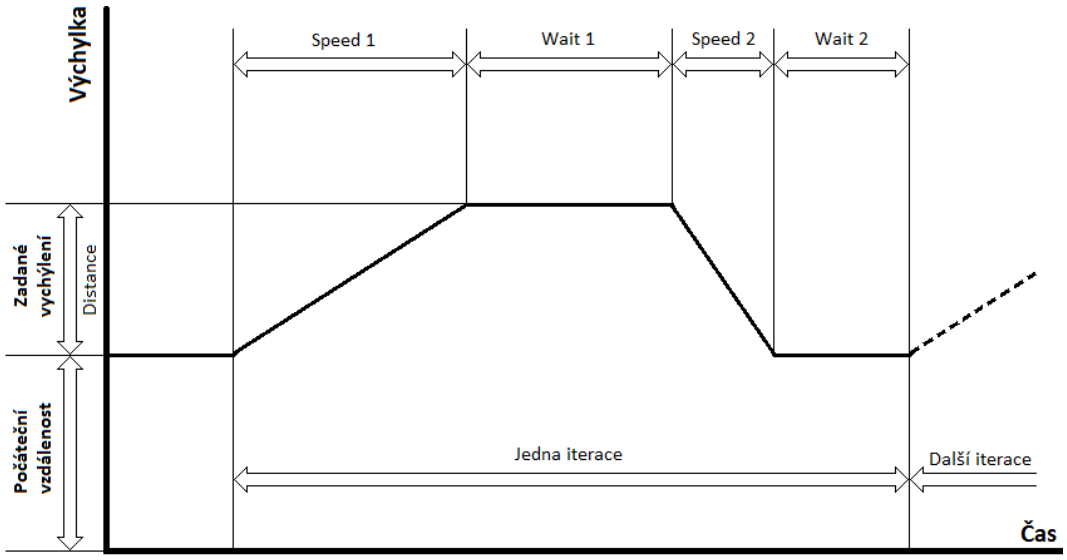
- Realizace „generálního stopu“
- Otestování komunikačního protokolu popsaného v kapitole 4.2.7.4
- Možnost použít aplikaci pro dlouhodobé testování v zátěži

5.2.2 Argumenty pro spuštění demonstrační aplikace

Demonstrační aplikace obsahuje 2 různé režimy činnosti. Výběr mezi nimi se provádí počtem zadaných argumentů. V případě chybných nebo chybného počtu argumentů se vypíše stručná nápověda k obsluze. Aplikace je uzpůsobena pro spuštění z příkazové řádky.

5.2.2.1 Argumenty aplikace pro program „pila“

- Port sériové linky – pouze číslo portu
- Amplituda – velikost výchylky motoru
- Rychlost 1 – rychlost dopředného pohybu
- Čekání 1 – délka prodlevy mezi dopřednou a zpětnou fází pohybu
- Rychlost 2 – rychlost zpětného pohybu
- Čekání 2 – délka prodlevy mezi zpětnou a dopřednou fází pohybu
- Počet iterací – celkový počet vygenerovaných průběhů



Obrázek 5.2: Funkcionalita realizovaná demonstračním programem „pila“

5.2.2.2 Argumenty aplikace pro program jednoduchého posuvu

- Port sériové linky – pouze číslo portu
- Amplituda – velikost výchylky motoru
- Rychlost – rychlost pohybu motoru

5.2.3 Poznatky z nasazení demonstrační aplikace

Tuto demonstrační aplikaci jsem používal přibližně 9 měsíců, během kterých jsem hledal chyby v realizované hardwarové části řídícího systému. Během testování se objevily tyto problémy:

- Nekorektní vykonávání instrukcí řídícího systému
- Nestabilita řídícího systému

U obou těchto problémů jsem předpokládal společnou příčinu, neboť jsem nebyl schopen chybu odhalit během simulace (Behavior). První problém se projevoval zahozením první instrukce MOVE po spuštění řídící jednotky, při jejím resetu, při použití instrukcí přerušení a obnovení činnosti. Příčinu se mi povedlo nalézt až po 8–mi měsících, kdy jsem byl schopen přesně identifikovat jevy vedoucí k tomuto chování. Příčinou tohoto chování byl špatně implementované chování registru, který syntézní nástroj interpretoval jinak než nástroj pro simulace. Řešením bylo poupravit zápis registru v jazyce VHDL tak, aby test simulací proběhl úspěšně i v režimu simulace po překladu na logickou strukturu (Post–Synthesis).

Druhým problémem bylo nepredikovatelné selhání funkčnosti zařízení. Podezření padlo na nízkou odolnost vůči všem typům rušení a dalších vnějších vlivů. Mezi pozorované jevy

patřilo vyřazení systému při doteku obsluhy na neuzemněné části zařízení. Ne všechny pozorované problémy byly zapříčeny výbojem statické elektřiny. Jednou z možných příčin mohlo být i pronikání elektromagnetického rušení do zapojení hardwarové části řídícího systému. Toto rušení se mohlo indukovat v příliš velkých proudových smyčkách v zapojení nebo pronikat z napájecích zdrojů a dalších částí systému skrze přívodní vodiče. U některých dosažených stavů systémů jsem nedokázal přijít na jinou možnou příčinu než SEU (Single Event Upset) [13].

5.2.4 Popis provozního prostředí

Původní provozní prostředí řídícího systému pro ovládání lineárního motoru a řídící jednotky ESMC–C2 je Laboratoř biomechaniky člověka Fakulty strojní. Tato laboratoř je umístěna v halových laboratořích, kde je velké množství přístrojů a strojního vybavení, které mají buď velký odběr nebo pro svou činnost používají výkonové DC/DC měniče. Tyto provozní podmínky nejsou doporučeny pro provoz lineárního motoru EZ Limo EZC6E030M–C s řídící jednotkou ESMC–C2, jak se lze dočíst v dokumentaci poskytované výrobcem [2] (kapitola 2.4.2).

Před zahájením procesu hledání a odstraňování chyb byl střední čas běhu systému přibližně jeden a půl hodiny. V případě kontaktu obsluhy bez zemnícího náramku na některé části řídícího systému byla pravděpodobnost zastavení řídícího systému přibližně 40%.

5.2.5 Možná opatření k odstranění problémů

Po konzultaci s členy katedry mikroelektroniky Fakulty elektrotechnické jsem stanovil možné způsoby odstranění těchto problémů. Řazení je provedeno podle složitosti realizace a ceny opatření:

1. Uzemnění všech částí zařízení
2. Stínění použitých vodičů nebo celého zařízení
3. Geometrické uspořádání pro omezení induktivní vazby mezi vodiči
4. Nahradit použité spínané zdroje lineárními
5. Kvalifikovaný návrh nové verze desky plošných spojů
6. Filtrace napájení pro všechny části zařízení

5.2.5.1 Uzemnění všech částí zařízení

Prvním krokem uzemnění všech elektrických a mechanických částí systému. Při revizi zapojení lineárního motoru jsem zjistil nepřipojenou zemnící svorku a kovový oplet přívodního kabelu končil přibližně 0,5 cm před samotným tělesem (kostrou) motoru. To bylo zapříčeno neodbornou manipulací s přívodními vodiči členy spřátelené katedry Anatomie a Biomechaniky, UK FTVS. V popsaných skutečností je patrné, že lineární motor a jeho kabeláž fungovala jako anténa o délce přes 2 metry.

5.2.5.2 Stínění použitých vodičů nebo celého zařízení

Druhým krokem bylo provést stínění datových a napájecích vodičů. Před uzemněním tělesa lineárního motoru jsem provedl orientační měření na osciloskopu. Sondu osciloskopu jsem připojil na kovovou část tělesa. Během pozorování často indukované napětí přesáhlo hranici 1 V. Vzhledem k rozhodovací úrovni řídící jednotky na úrovni 1,65 V je pravděpodobné, že některé zákmity mohou tuto hranici překonat.

Při nahrazování nestíněné kabeláže jsem používal kabeláž STP CAT5e, která se používá pro rozvody Ethernetu. Tento kabel se vyznačuje 4 páry kroucených vodičů a kovovou fólií chránících všechny vodiče. Z každého páru je jeden vodič vyhrazen pro signál a druhý pro jeho stínění. Stínící fólie a stínící vodiče byly vzájemně spojeny a připojeny k signálové zemi.

5.2.5.3 Geometrické uspořádání pro omezení induktivní vazby mezi vodiči

Třetím krokem kromě přidání stínění k vodičům je omezení induktivní, kapacitní a galvanické vazby mezi jednotlivými vodiči. Tyto vazby jsou silnější, čím je větší frekvence rušení. Je tedy důležité neumisťovat vodiče s rušením do blízkosti vodičů bez výskytu rušení. Při porušení tohoto doporučení hrozí nekorektní funkce zařízení.

5.2.5.4 Nahradit použité spínané zdroje lineárními

Čtvrtým krokem je nahradit používané spínané napájecí zdroje zdroji lineárními. Spínaný zdroj pracuje na principu spínání výkonového tranzistoru na vysoké frekvenci. Rychlými změnami je na výstupní cívce indukováno cílové napětí, zároveň je indukčností udržován výstupní proud [19].

Problémem těchto zdrojů je pronikání pracovního kmitočtu do výstupního napětí. Kromě možnosti filtrace výstupního napětí je možné použít napájecí zdroje s lineárním napěťovým regulátorem. Lineární zdroj je složen z tranzistoru, který omezuje výstupní proud a napětí, a analogové zpětné vazby, která řídí otevírání tranzistoru. V tomto typu napájecího zdroje se tedy nevyskytuje žádný pracovní kmitočet.

5.2.5.5 Kvalifikovaný návrh nové verze desky plošných spojů

Deska plošných spojů ve verzi 2, popsaná v kapitole 4.3.3, je realizována jako dvouvrstvá DPS. Tento typ konstrukce má horší předpoklady pro odolnost vůči elektromagnetickému než více vrstvé desky plošných spojů. U čtyř– a více–vrstvých DPS jsou některé vrstvy vyhrazeny jen pro rozvody napájení. Tyto vrstvy pak fungují jako deskový blokovací kondenzátor schopný filtrovat vysokofrekvenční rušení [28].

Dalším problémem DPS ve verzi 2 byly velké plochy proudových smyček. Ty mohou vysílat a přijímat elektromagnetické rušení. Použitím více vrstev DPS bude mnohem snažší se vyhnout ostatním cestám a výsledné vodivé cesty by tak měly být kratší. Tím se zmenší plocha proudových smyček a náchylnost zařízení vůči elektromagnetickému rušení [28].

5.2.5.6 Filtrace napájení pro všechny části zařízení

Posledním možným způsobem odstranění rušení, které proniká do hardwarové části řídícího systému, je instalace vysokofrekvenčního filtru pro hlavní napájecí přívod 230 V AC. Tento VF filtr bývá často integrován do přepěťové ochrany III. třídy. Tento typ ochrany je zároveň doporučen výrobcem lineárního motoru EZ Limo EZC6E030M–C a jeho řídící jednotky ESMC–C2, jak je řečeno v kapitole 2.4.2. Bohužel zařízení s tímto typem funkce jsem neměl k dispozici z důvodu vysoké pořizovací ceny.

5.2.6 Výsledky dosažené realizací jednotlivých opatření

Na základě jednotlivých doporučení pro odstranění problémů s funkčností zařízení jsem postupně realizoval body 1 — 4 ze seznamu 5.2.5. V popsaném prostředí laboratoře (kapitola 5.2.4) se mi povedlo zvýšit dobu provozu řídícího systému ze střední doby 1,5 hodiny na přibližně 1,5 dne. Dále se mi již nepovedlo vyvolat zastavení řídícího systému pomocí výboje statické elektřiny.

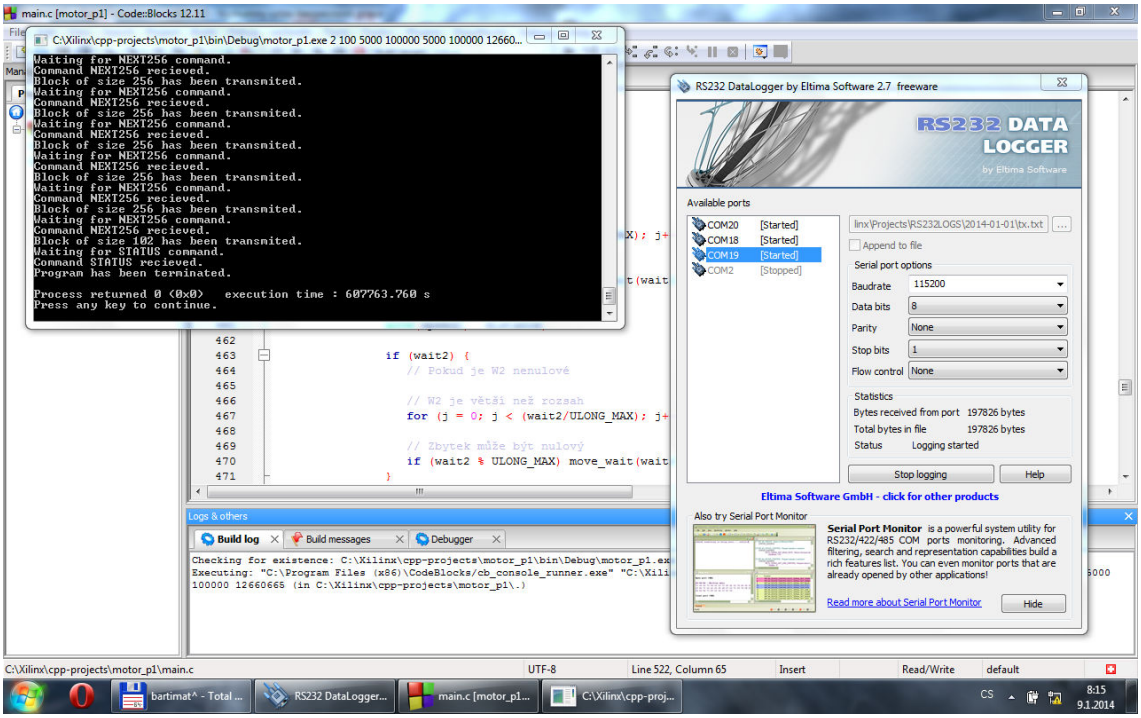
Doby běhu okolo 1,5 dne jsem obvykle dosáhl za následujících podmínek:

- Systém byl spuštěn během konce pracovního dne, kdy většina strojního vybavení byla vypnutá.
- Systém byl v provozu druhý den ráno, kontrolu provedli pracovníci Laboratoře biomechaniky člověka při vstupu do laboratoře.
- Systém nebyl v provozu v době polední pauzy. V tuto dobu běžela většina strojního vybavení v prostoru halových laboratoří.

Na základě těchto výsledků jsem došel ke dvěma možným příčinám nefunkčnosti zařízení. První je stále nedostatečná odolnost zařízení vůči elektromagnetickému rušení a nedodržení zásad EMC. Druhým možným důvodem je chyba v obslužné aplikaci. Pro vyloučení chyby v řídící aplikaci jsem odpojil vývojovou desku s FPGA, kterou jsem si odnesl na provedení testů k sobě domů. Poté jsem spustil obslužnou aplikaci a pozoroval, zda dojde k chybě. Celková doba testování byla 3 týdny. Poslední 2 testy byly spuštěné na dobu odpovídající jednomu týdnu. Oba tyto testy (obrázek 5.3) skončily bez jakékoliv chyby.

Pro sledování činnosti systému jsem připojil 3 převodníky z USB na UART, kterými jsem sledoval vstupy a výstupy rozhraní UART. Na obrázku 5.3 je zobrazena hodnota 197 826 bytů vyslaných systémem. Po odečtení 6 bytů na zahájení a ukončení programu odpovídá tato hodnota vykonání 50 641 920 instrukcím typu MOVE a WAIT. Stejným způsobem odpovídal počet odeslaných bytů do řídícího systému.

Na základě těchto testů jsem vyloučil chybu v obslužné aplikaci a případnou fatální chybu v koncepci řídícího systému. Proto jsem se rozhodl celý řídící systém přestěhovat do prostor, kde jsem předpokládal žádné nebo malé elektromagnetické rušení. Celý systém byl přestěhovat do pracovny diplomantů katedry číslicového návrhu Fakulty informačních technologií. Tato místnost je vybavena přepěťovými ochranami s VF filtrem popsaných v kapitole 2.4.2. V tomto novém provozním prostředí byly spuštěny testy, které trvaly přibližně 4 týdny. Poslední 2 testy probíhaly vždy 5 pracovních dnů a proběhly bez chyb.



Obrázek 5.3: Stavy použitých aplikací po dokončení týdenního testu

5.3 Použité hardwarové a softwarové nástroje při testování

Pro testování simulací jsem využíval simulátor Mentor Graphic ModelSim ve verzi 10.0c. Pro vývoj v jazyce C jsem využíval vývojové prostředí Code::Blocks. Pro rozhraní sériové linky jsem použil multiplatformní knihovnu „RS–232 for Linux and Windows“ [25]. Pro účely řídícího systému je nutné aplikaci napsat paralelně, aby bylo možné nezávisle reagovat na události pocházející z hardwarové části řídícího systému a na události pocházející od obsluhy. Z tohoto důvodu jsem použil knihovnu realizující POSIXová vlákna (pthread) [15], která zajistí paralelnost aplikace.

K testování stability řídícího systému jsem využíval převodníky z rozhraní USB na UART. Sledování jsem prováděl aplikací Eltima RS232 DataLogger [1].

5.4 Závěry z testování

Na základě těchto testů je možné s velkou pravděpodobností prohlásit, že nekorektní funkčnost celého zařízení je způsobena nedodržením požadavků na elektromagnetickou kompatibilitu (EMC). Tyto chyby celého zařízení by měly být odstraněny ve třetí verzi desky plošných spojů, která je popsána v kapitole 4.3.5 a v současné době je ve stádiu návrhu a realizace. Po dokončení této revize desky plošných spojů bude nutné obdobným způsobem DPS otestovat [22].

Kapitola 6

Závěr

V této práci jsem na základě analýzy lineárního motoru EZ Limo EZC6E030M–C a jeho řídící jednotky ESMC–C2 navrhl a realizoval nový řídící systém. Tento systém je rozdělen na několik různých úrovní podle možnosti jejich abstrakce. Vrstvy systému jsou rozděleny podle způsobu realizace na vrstvy softwarové a vrstvy hardwarové.

Vrstvy řídícího systému implementované v softwaru realizují funkci matematického aparátu nutnou pro účinné řízení lineárního motoru a funkci komunikačního protokolu s vrstvami realizovanými v hardwaru. Vrstvy řídícího systému realizované v hardwaru realizují systém nové řídící jednotky implementovaný v čipu FPGA. Vrstva řídícího systému realizovaná v FPGA je plně predikovatelná a má garantované časy obsluh pro každou událost řídícího systému. Pro správnou funkčnost jsem navrhl a vyrobil desku plošných spojů, která představuje nejnižší vrstvu řídícího systému. Poslední vrstvou realizovanou v hardwaru i softwaru je funkce komunikačního protokolu, která tvoří přechod mezi softwarovou a hardwarovou doménou návrhu.

Řídící systém a všechny jeho části byly podrobeny rozsáhlému testování pomocí simulace, s využitím demonstračních programů a měřících přístrojů pro elektrotechnická měření. Systém komunikuje s prostředím MATLAB a vykonává všechny zadané funkce za předpokladu zajištění odpovídajících provozních podmínek. V současné době řídící systém a jeho části nesplňují předpisy týkající se elektromagnetické kompatibility (EMC).

Pro nasazení do plného provozu je nutné použít některé z navrhovaných úprav na zlepšení elektromagnetické kompatibility, například instalace vysokofrekvenčního filtru pro napájecí napětí a realizace nové desky plošných spojů se zvýšenou odolností vůči elektromagnetickému rušení.

Pro základní měření v biomechanice jsem napsal demonstrační aplikaci „pila“, která vykonává pohyb motoru podle zadaných parametrů (rychlost, vzdálenost, prodlevy).

Jako možná pokračování práce se nabízí přepracování hardwarové části řídícího systému za účelem zvýšení jeho odolnosti vůči elektromagnetickému rušení. Druhou možností pokračování práce je realizace pokročilé obslužné aplikace pro řízení činnosti lineárního motoru, která využije dosud nevyužitou část instrukčního souboru řídícího systému.

Literatura

[1] Eltima Software, 2010. Dostupné z: <http://www.eltima.com>.

[2] EZC Controller Manual. Dostupné z: <http://www.oriental-motor.co.uk/media/files/hl-17082-5e-ezc-controller-manual-en.pdf>. 2011.

[3] Quadrature Decoder, 2013. Dostupné z: <http://www.fpga4fun.com/QuadratureDecoder.html>.

[4] Systems Design Using FPGAs, 2008-09-25. Dostupné z: <http://www.ami.ac.uk/courses/ami4460_fpga/u02/>.

[5] A single master I2C tutorial. [online], 2005. Dostupné z: <http://www.best-microcontroller-projects.com/i2c-tutorial.html>.

[6] LogiCORE IP FIFO Generator v9.3. Dostupné z: <http://www.xilinx.com/support/documentation/ip_documentation/fifo_generator/v9_3/pg057-fifo-generator.pdf>. 2012-12-18.

[7] Mbed: QEI. Development Platform for Devices, 2010. Dostupné z: <https://mbed.org/cookbook/QEI>.

[8] Papilio Pro, 2013. Dostupné z: <http://papilio.cc/index.php?n=Papilio.PapilioPro>.

[9] Spartan-3 FPGA Family Data Sheet. Dostupné z: <http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf>. 2013-06-27.

[10] Spartan-3E FPGA Family Data Sheet. Dostupné z: <http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf>. 2013-07-19.

[11] Xilinx Spartan-3E FPGA Sample Pack User's Guide. Dostupné z: <http://www.xilinx.com/products/boards/s3esamplepack/files/S3Euserguide.pdf>. 2005.

[12] SALTEK DA-275 DF 16. Dostupné z: <http://www.saltek.eu/files/vyrobky/DA-275%20DF%2016.pdf>. 1995.

[13] Single Event Upsets – Xilinx Inc., 2014. Dostupné z: <http://www.xilinx.com/products/quality/single-event-upsets.htm>.

[14] Saleae Logic analyzer, 2012. Dostupné z: <https://www.saleae.com/logic>.

[15] The Single UNIX Specification – pthread.h, 1997. Dostupné z: <http://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread.h.html>.

[16] DAVID O. MCLAURIN, P. Automated Discretization of Digital Curves through Local or Global Constrained Optimization, 2012. Dostupné z: <http://www.dtic.mil/ndia/2012physics/Wednesday14973_McLaurin.pdf>.

[17] GOFTON, P. W. *Sériová komunikace*. Praha : Grada, 1. vyd. edition, 1994.

[18] IRAZABAL, J.-M. – BLOZIS, S. AN10216-01. Dostupné z: <http://www.nxp.com/documents/application_note/AN10216.pdf>. 2003-03-24.

[19] JAN BABčANíK ING., J. Spínané zdroje, 2007-05-02. Dostupné z: <http://www.hw.cz/teorie-a-praxe/spinane-zdroje.html>.

[20] ING. ZAPLATíLEK, P. – ING., M. T. Elektrorevue, 2009. Dostupné z: <http://www.elektrorevue.cz/file.php?id=200000342-8a24e8b1ec>.

[21] KOVáč, D. – KOVáčOVá, I. – KAňUCH, J. *EMC z hlediska teorie a aplikace*. Praha : BEN - technická literatura, 1. vyd. edition, 2006.

[22] MATTHES, W. *Hledání a ostraňování poruch*. Ostrava : HEL, 1. vyd. edition, 2001.

[23] V, R. S. K. AN61345. Dostupné z: <http://www.cypress.com/?rID=43046>. 2014-03-19.

[24] VACULíKOVá, P. *Elektromagnetická kompatabilita elektrotechnických systémů*. Praha : Grada, 1. vyd. edition, 1998.

[25] BEELEN, T. RS-232 for Linux and Windows, 2011. Dostupné z: <http://www.teuniz.net/RS-232/>.

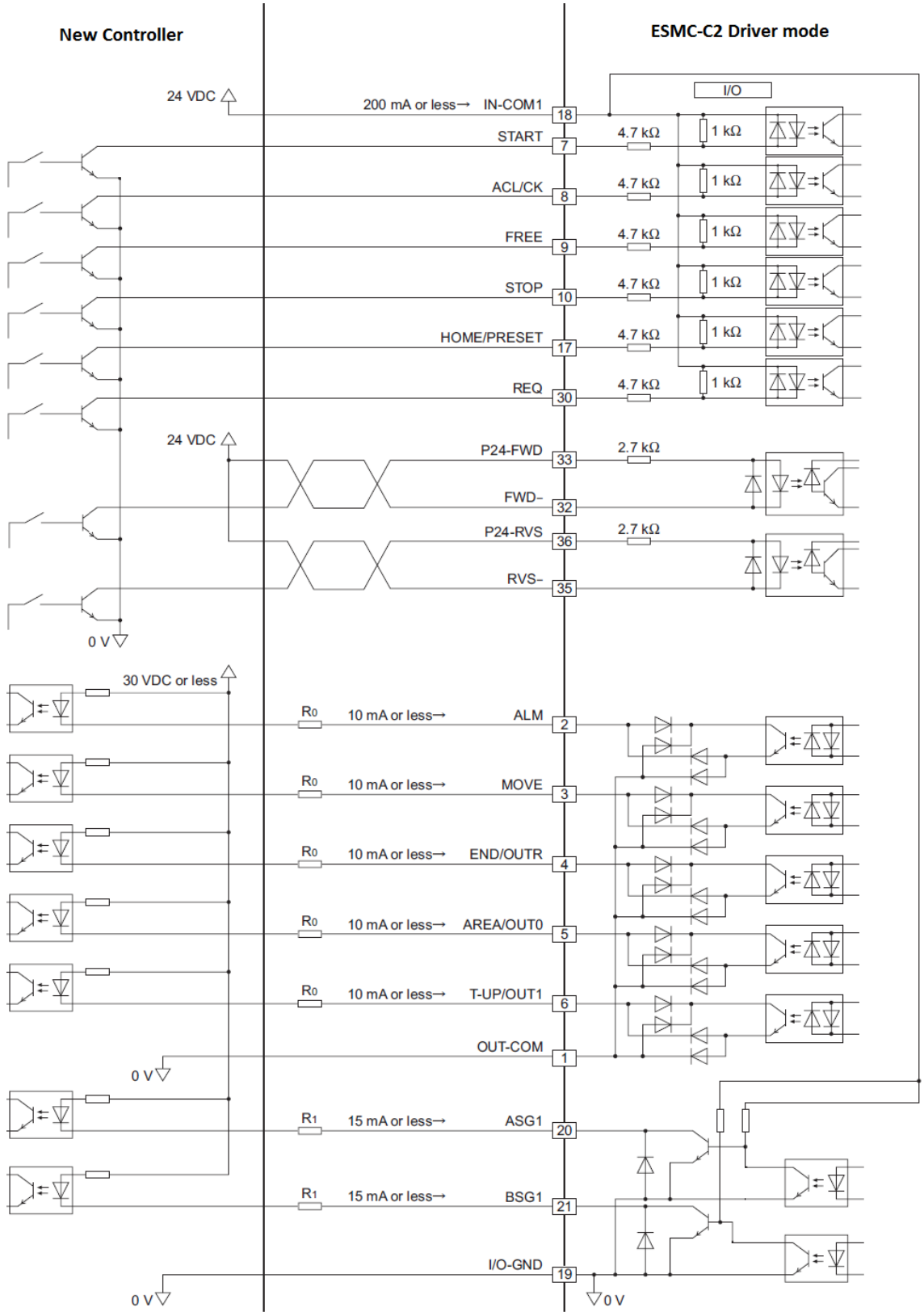
[26] ZAPLATíLEK, K. *MATLAB pro začátečníky*. Praha : BEN - technická literatura, 2. vyd. edition, 2005.

[27] ZáHLAVA, V. *OrCAD 10*. Praha : Grada, vyd. 1. edition, 2004.

[28] ZáHLAVA, V. *Návrh a konstrukce desek plošných spojů*. Praha : BEN - technická literatura, 1. vyd. edition, 2010.

Příloha A

Elektronické zapojení řídící jednotky ESMC–C2



Obrázek A.1: Elektronické zapojení řídící jednotky ESMC–C2 a nové řídící jednotky

Příloha B

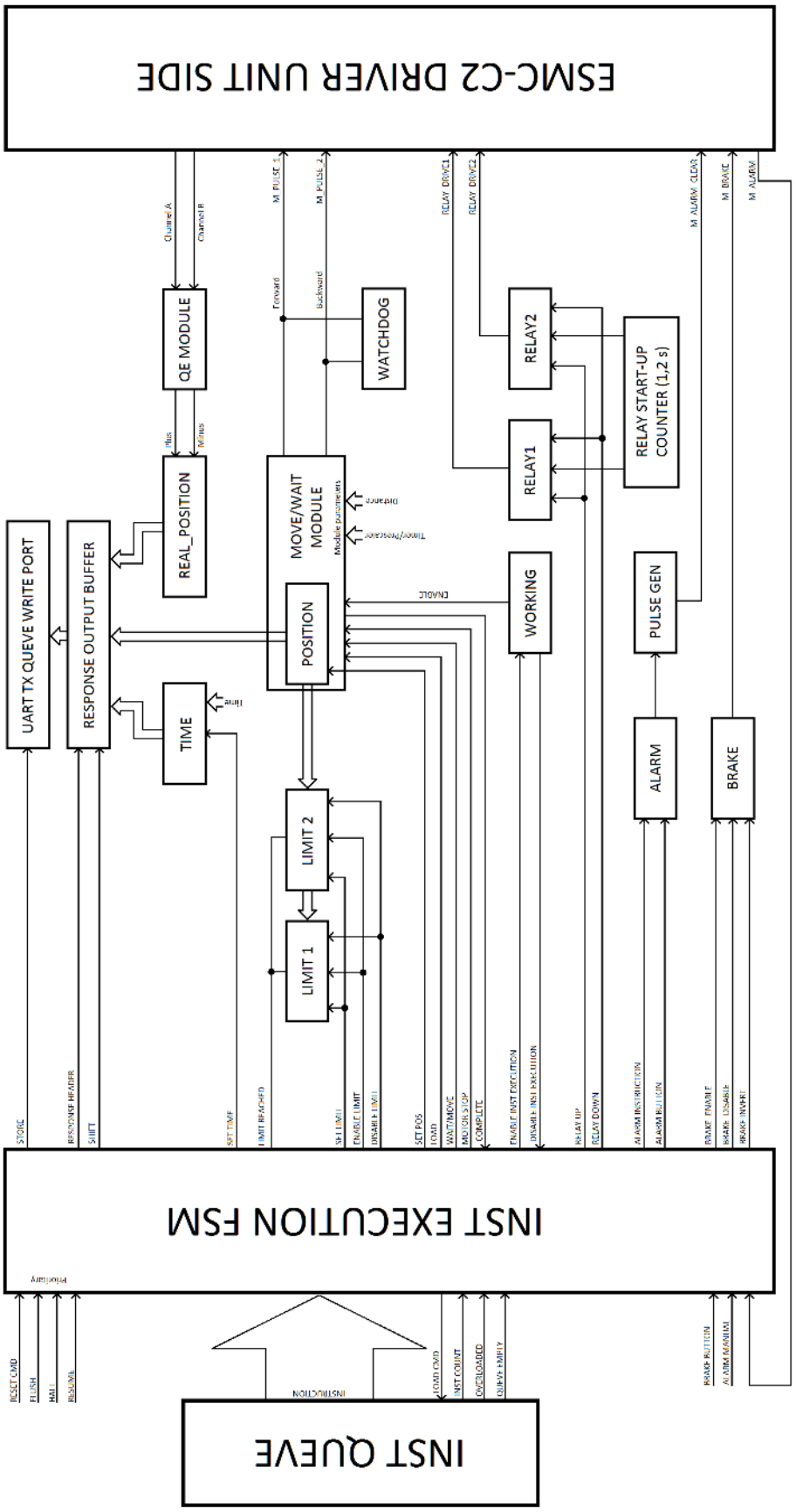
Instrukční soubor hardwarové části řídícího systému

Instrukční soubor hardwarové části řídícího systému					
Instrukce	Slovní popis	Binární reprezentace	Operand č.1	Operand č.2	Prioritní
NOP	Prázdná instrukce	0000 0000	Není	Není	Ne
MOVE_WAIT	Čekání mezi instrukcemi	1000 0001	Délka čekání v hodinových taktech	Není	Ne
MOVE	Pohyb motoru	1000 0000	Interval mezi 2 kroky motoru	Poček kroků motoru	Ne
HALT	Zastavení činnosti	0000 0001	Není	Není	Ano
FLUSH	Zastavení činnost a vymázání prováděného programu	0000 0011	Není	Není	Ano
RESUME	Zahájení/obnovení činnosti	0000 0010	Není	Není	Ano
RESETCMD	Restart celé řídící jednotky	0000 0100	Není	Není	Ano
SET_POS	Nastavení výchozí pozice	0100 0000	Výchozí pozice	Není	Ne
READ_POS	Přečtení pozice	0100 0001	Není	Není	Ne
READ_R_POS	Přečtení pozice získané z kvadraturního enkodéru	0100 0010	Není	Není	Ne
SET_LIMIT_1	Ohraničení dovoleného prostoru	0001 0000	Hranice prostoru	Není	Ne
DISEN_LIMIT_1	Vypnutí první hranice	0001 0001	Není	Není	Ne
EN_LIMIT_1	Zapnutí první hranice	0001 0010	Není	Není	Ne
SET_LIMIT_2	Ohraničení dovoleného prostoru	0001 0011	Hranice prostoru	Není	Ne
DISEN_LIMIT_2	Vypnutí první hranice	0001 0100	Není	Není	Ne
EN_LIMIT_2	Zapnutí první hranice	0001 0101	Není	Není	Ne
BRAKE_EN	Aktivace magnetické brzdy	0001 0110	Není	Není	Ne
BRAKE_DIS	Deaktivace magnetické brzdy	0001 0111	Není	Není	Ne
READ_TIME	Přečtení času jednotky	0010 0001	Není	Není	Ne
SET_TIME	Nastavení času jednotky	0010 0000	Výchozí čas	Není	Ne
STATUS_REQ	Žádost o stavové informace	0001 1000	Není	Není	Ne
ECHO	Odpověď zařízení	1100 0011	Není	Není	Ne
RELAY1_DOWN	Rozepnutí relé 1	1110 0000	Není	Není	Ne
RELAY1_UP	Sepnutí relé 1	1110 0001	Není	Není	Ne
RELAY2_DOWN	Rozepnutí relé 2	1110 0010	Není	Není	Ne
RELAY2_UP	Sepnutí relé 2	1110 0011	Není	Není	Ne
ALARM_CLEAR	Vymazání alarmu na řídící jednotce ESMC–C2	1010 0101	Není	Není	Ne

Tabulka B.1: Instrukční soubor hardwarové části řídícího systému

Příloha C

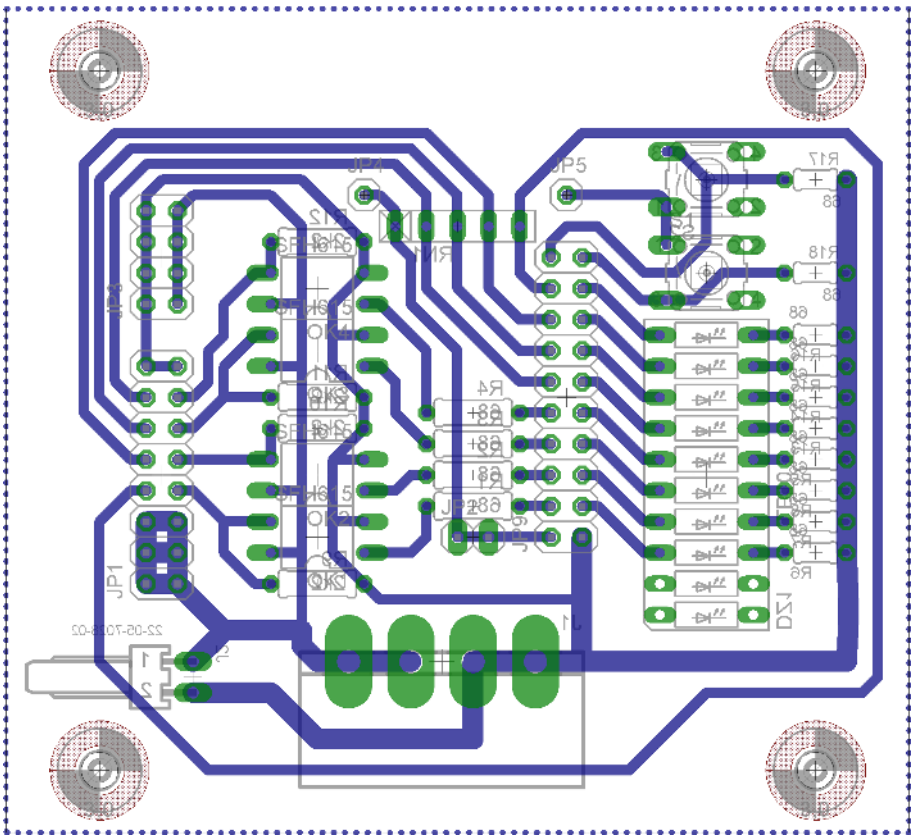
Vnitřní zapojení hlavního modulu pro provádění instrukcí



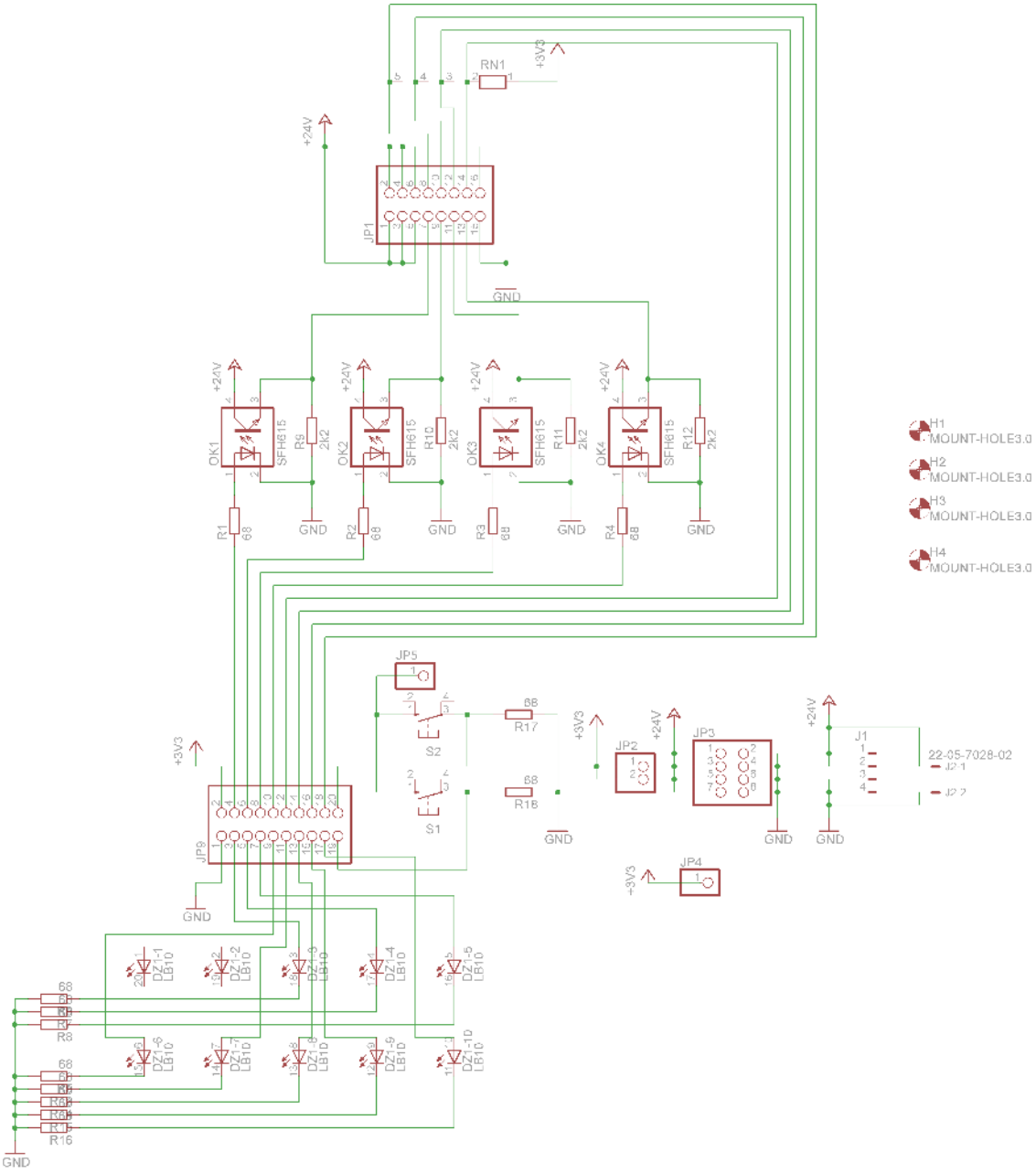
Obrázek C.1: Vnitřní zapojení hlavního modulu pro provádění instrukcí – zvětšené

Příloha D

Deska plošných spojů pro zapojení první verze



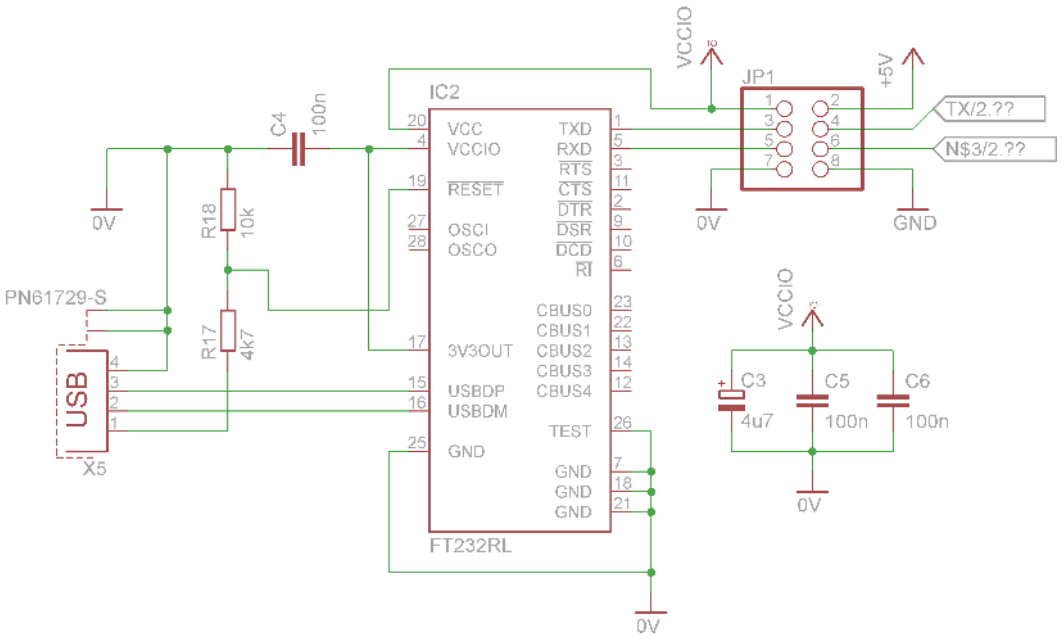
Obrázek D.1: Výsledná podoba desky plošných spojů pro první verzi zapojení



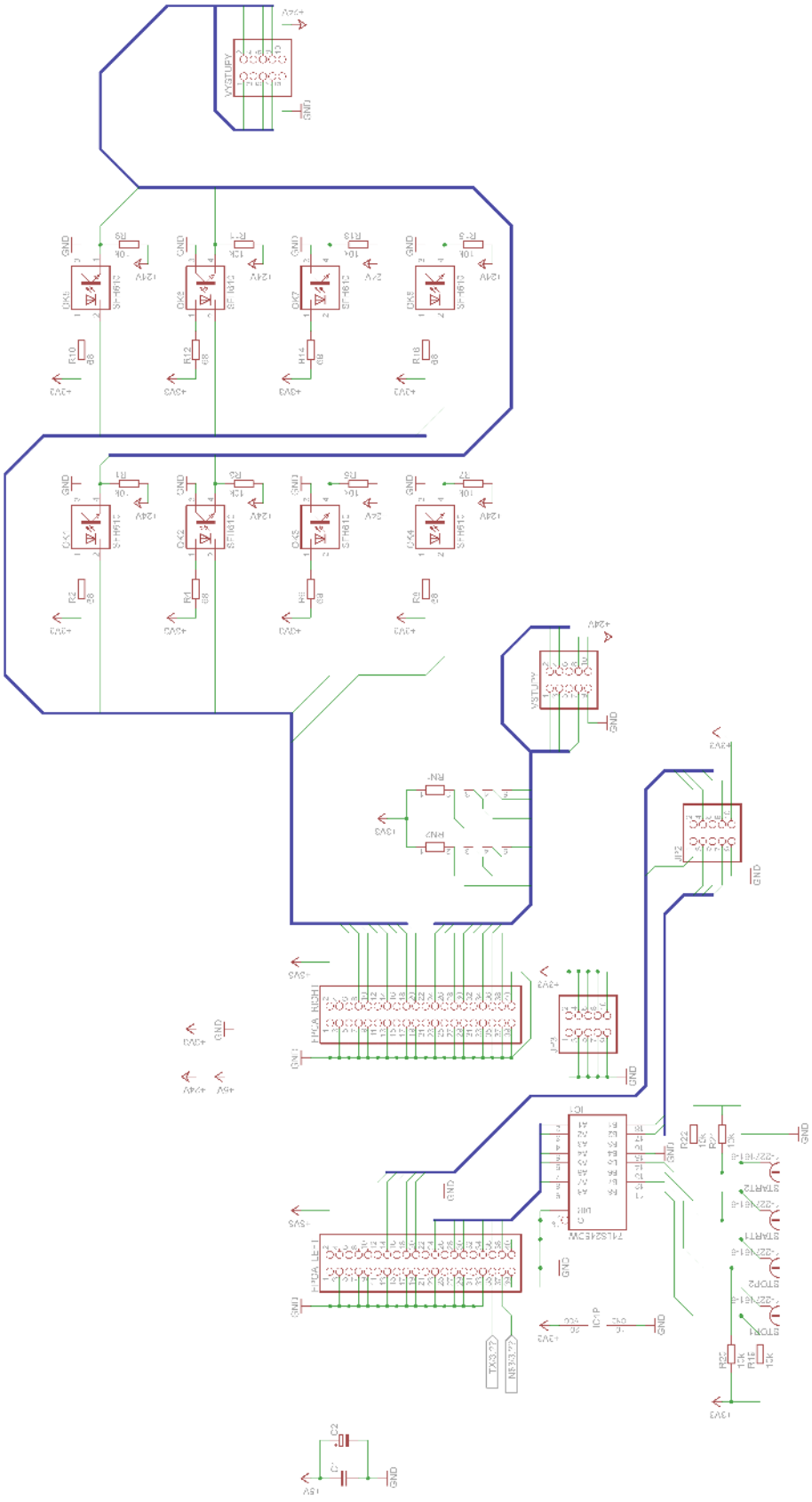
Obrázek D.2: Výsledné schéma pro první verzi zapojení

Příloha E

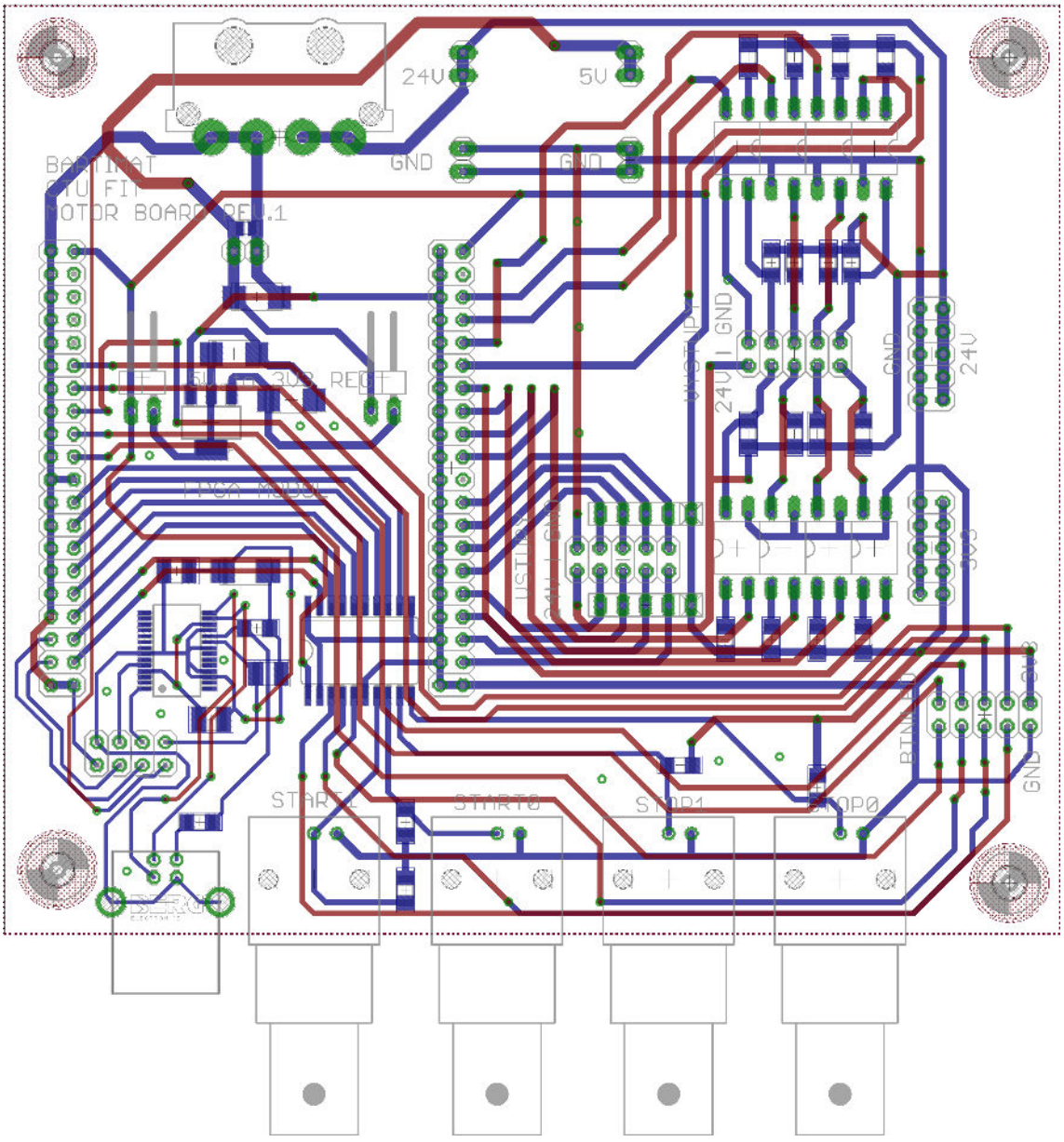
Deska plošných spojů pro zapojení druhé verze



Obrázek E.1: Zapojení obvodu FTDI FT232RL



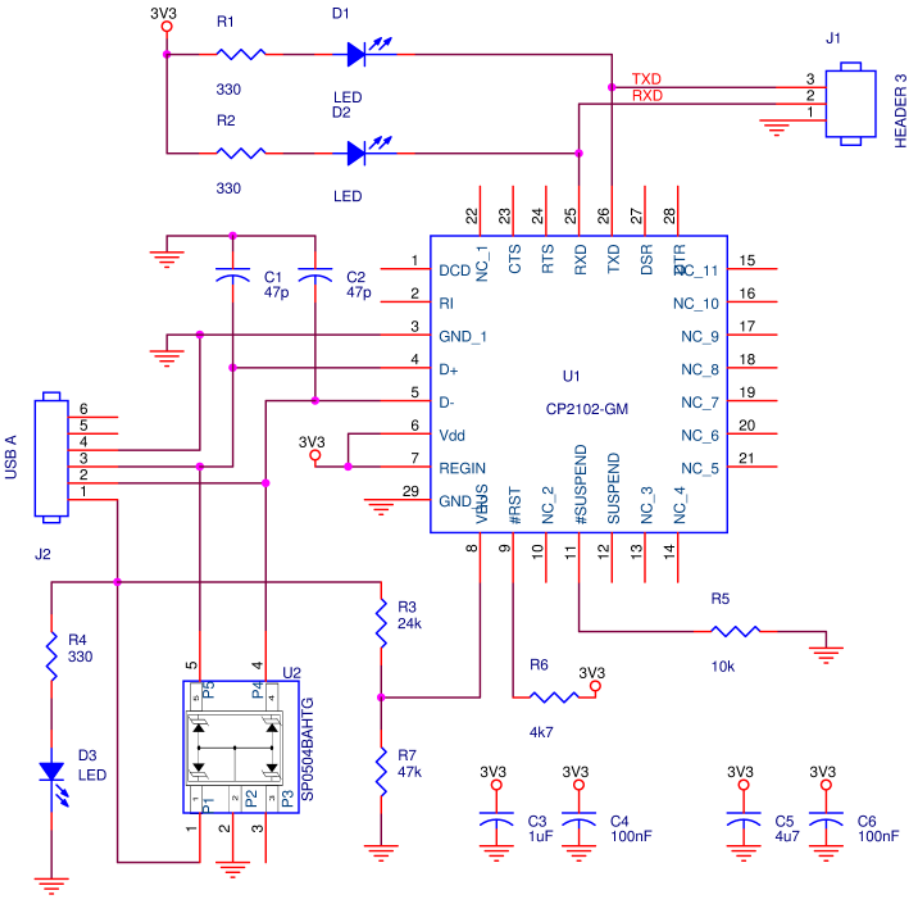
Obrázek E.2: Zapojení optočlenů, vývojové desky FPGA a DC/DC měniče.



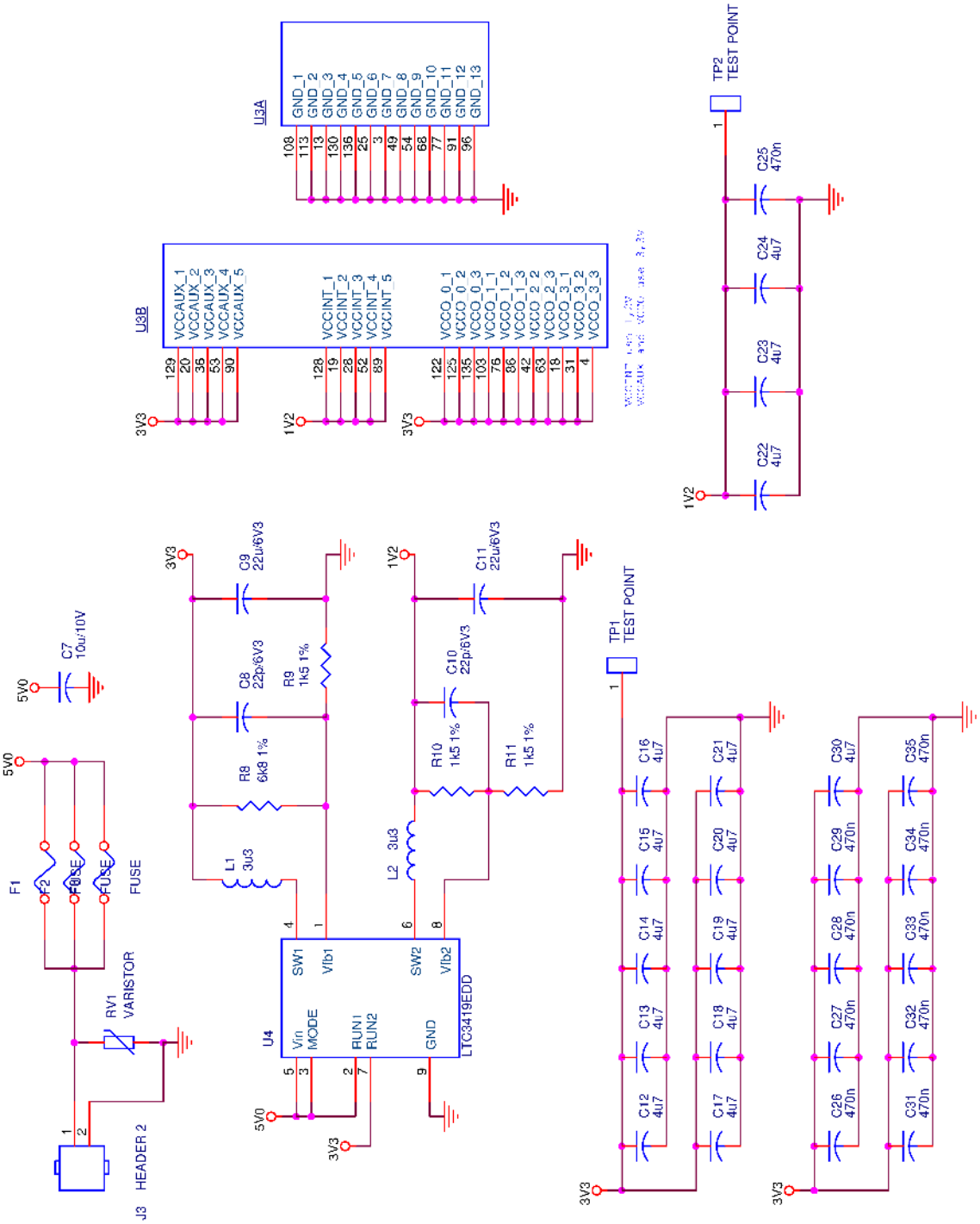
Obrázek E.3: Vzhled desky plošných spojů pro druhou verzi zapojení

Příloha F

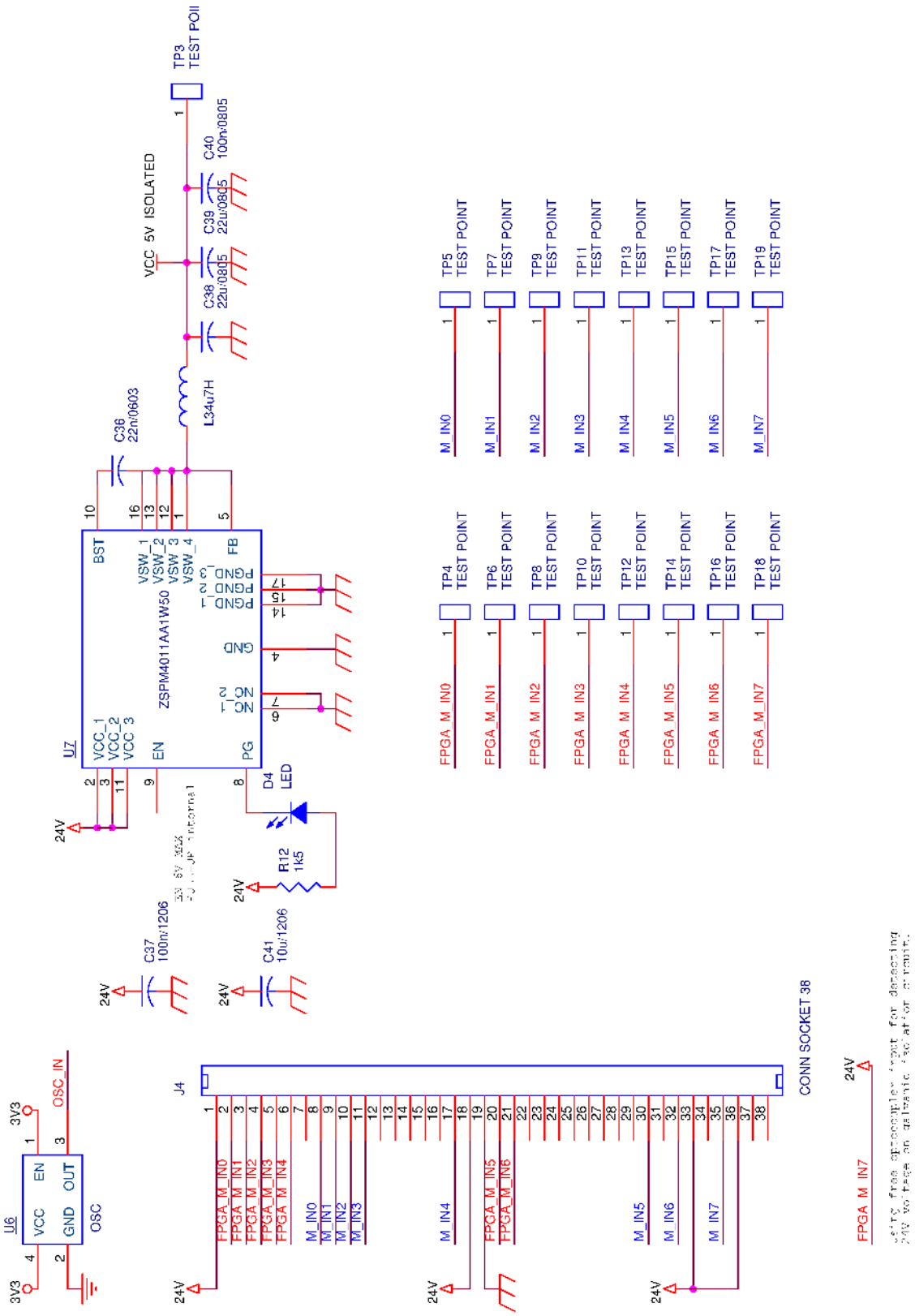
Zapojení desky plošných spojů – třetí finální verze



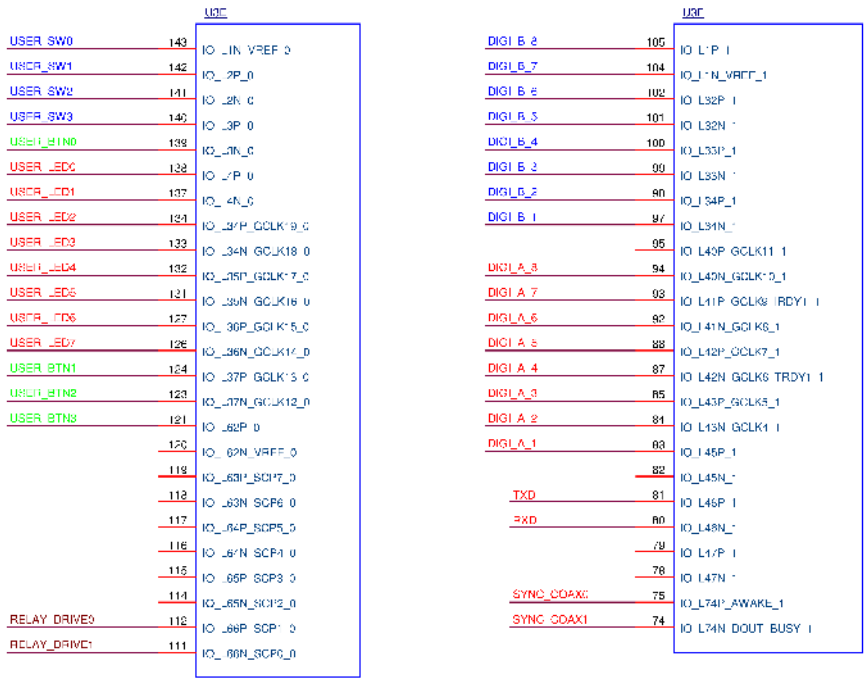
Obrázek F.1: Zapojení USB – UART převodníku s čipem CP2102



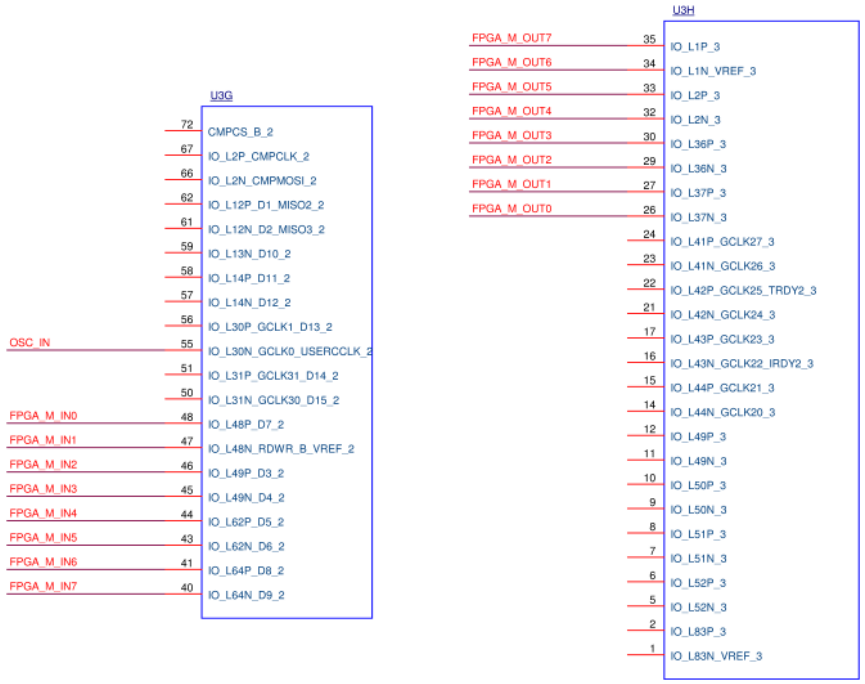
Obrázek F.2: Napájecí kaskáda FPGA a blokovací kondenzátory



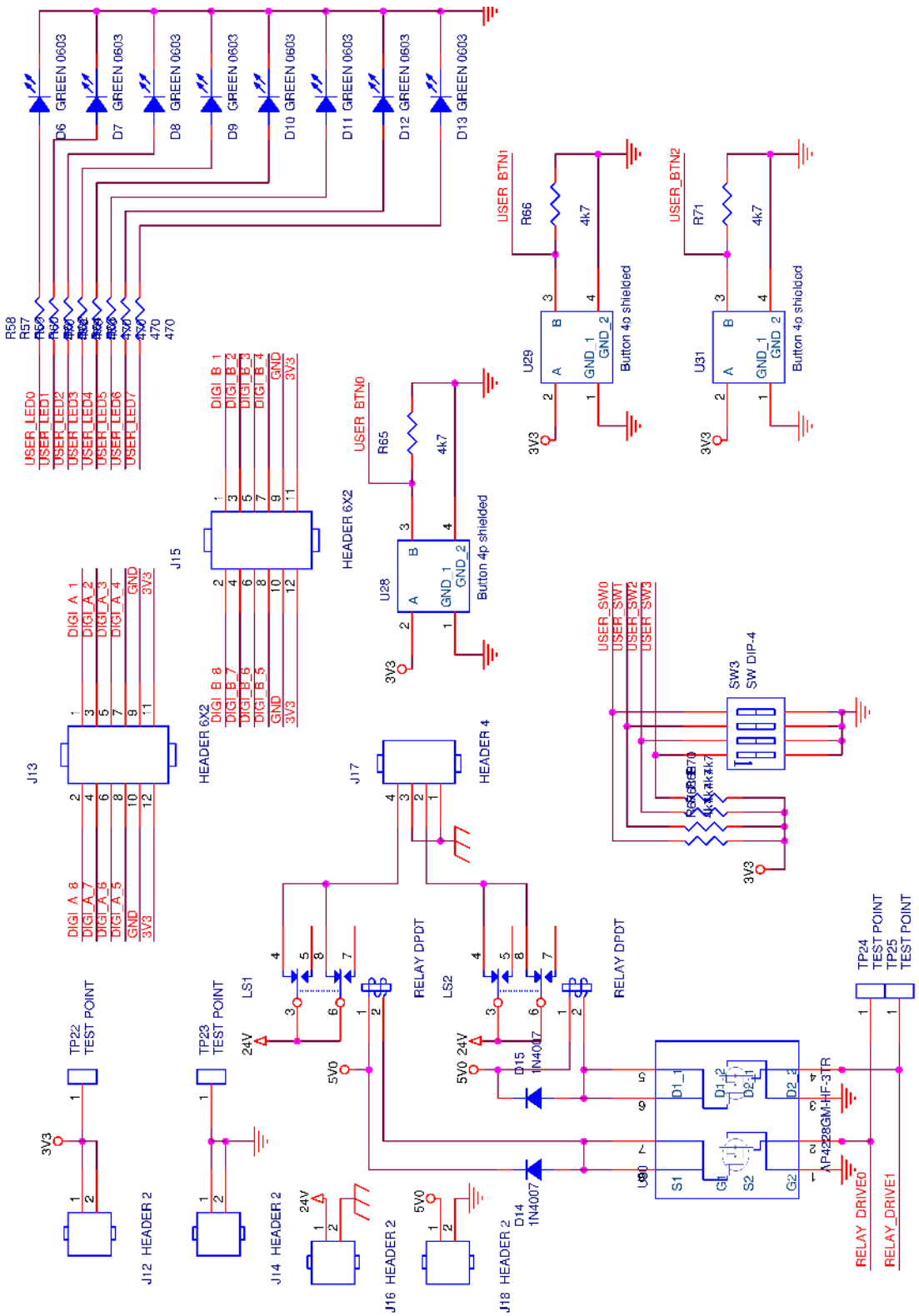
Obrázek F.3: Pomocný DC/DC měnič pro napájení optočlenů, oscilátor, systémový konektor k původní řídící jednotce ESMC–C2, měřící body



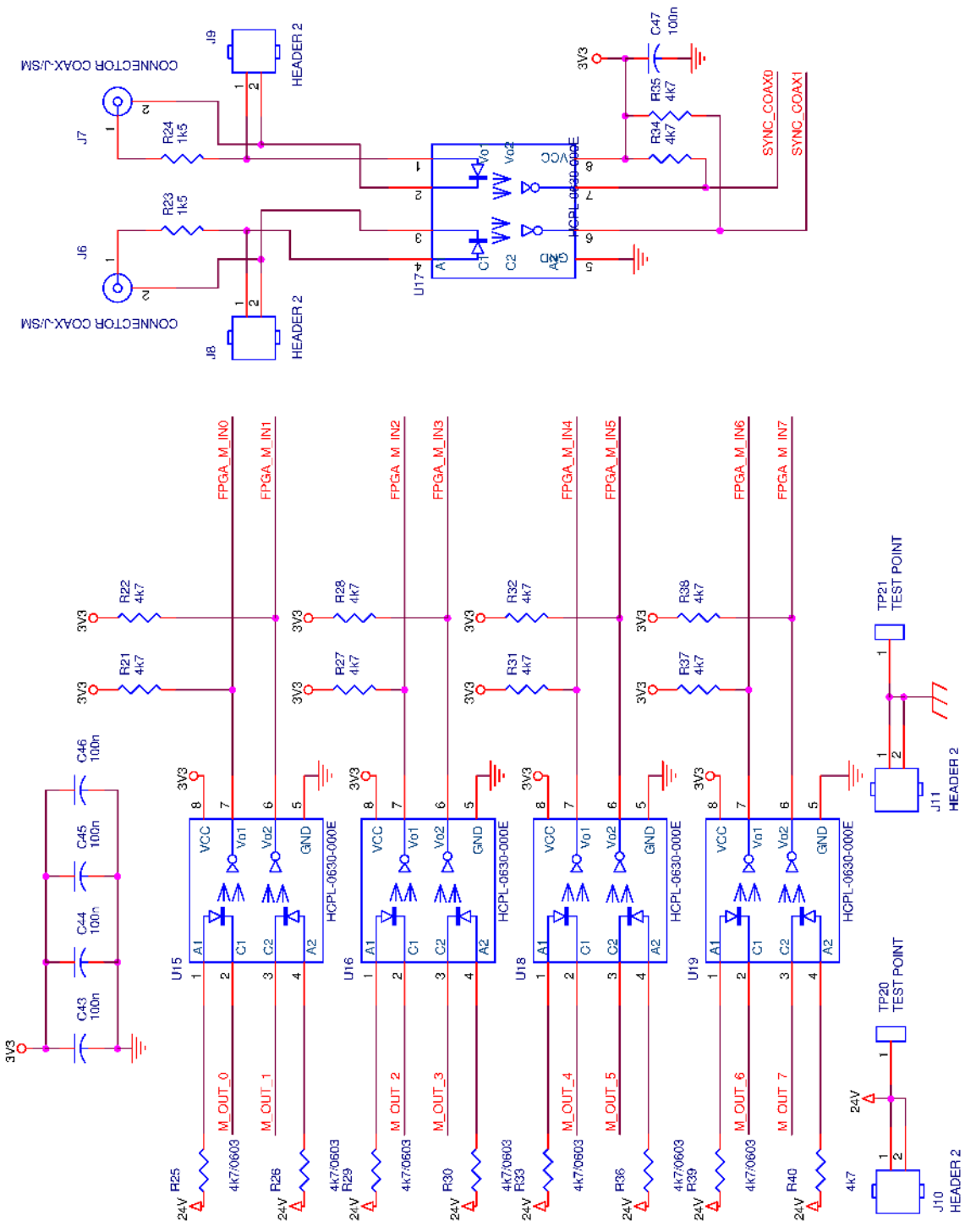
Obrázek F.4: Zapojení I/O BANK 0 a 1 čipu Spartan-6



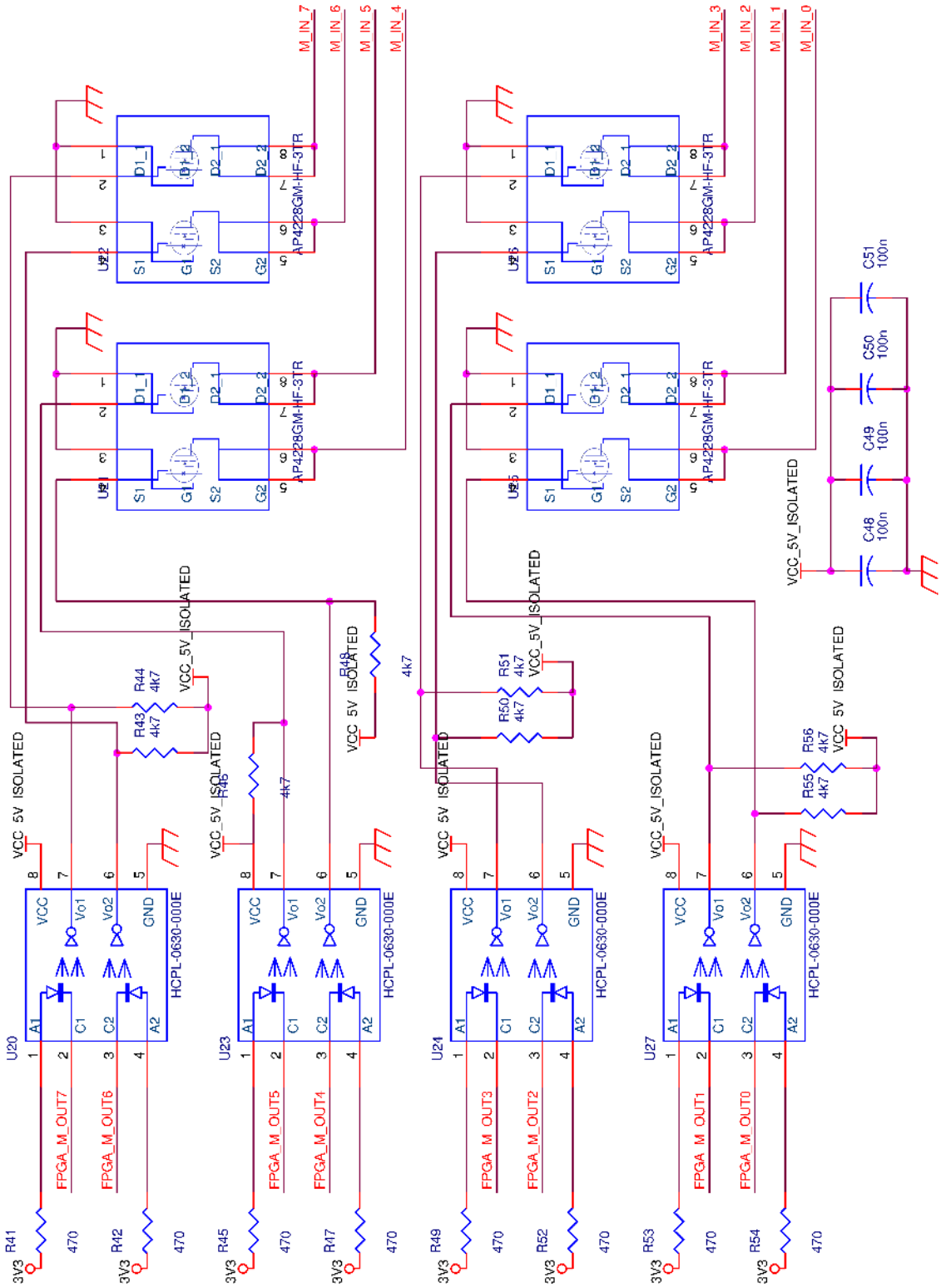
Obrázek F.5: Zapojení I/O BANK 2 a 3 čipu Spartan-6



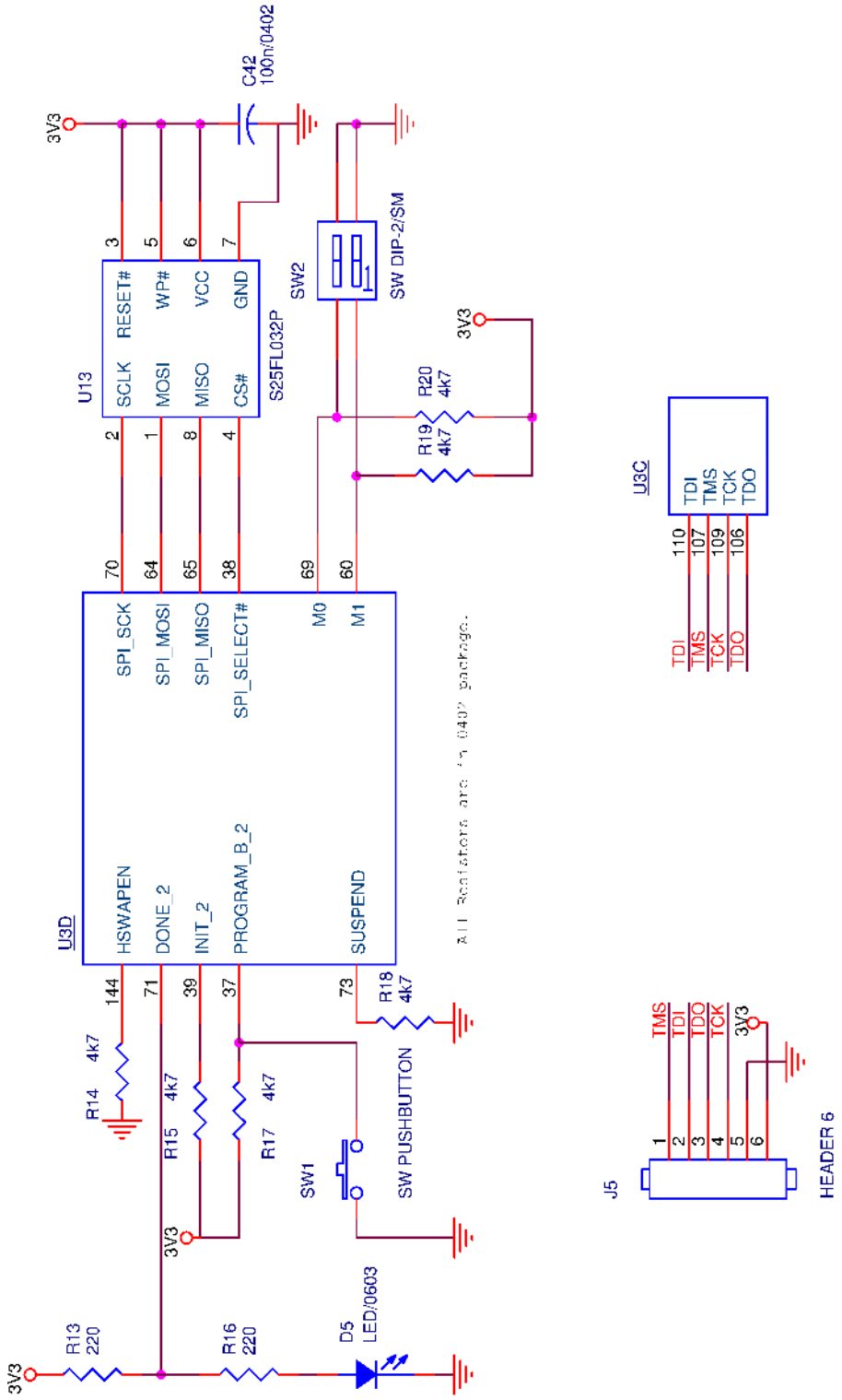
Obrázek F.6: Zapojení prvků pro ladění (tlačítka, LED, pinové lišty), obsluha relé



Obrázek F.7: Zapojení vstupních signálových portů s napěťovou konverzí pro FPGA



Obrázek F.8: Zapojení výstupních signálových portů s napěťovou konverzí pro cílové zařízení



Obrázek F.9: Zapojení systémových pinů FPGA, Flash PROM pro uložení FPGA bitstreamu, JTAG konektor

Příloha G

Seznam použitých zkratek

AC Alternating Current

A/D Analog – Digital

ARM Advanced RISC Machines

CPLD Complex Programable Logic Device

D/A Digital – Analog

DC Direct Current

DCM Digital Clock Managment

DSP Digital Signal Processor

DIN Deutsches Institut für Normung

DSP Deska plošných spojů

EMC Electromagnetic compatibility

FPGA Field-Programmable Gate Array

FSM Finite–state machine

I^{2}C Inter-Integrated Circuit

IP Core Intellectual Property Core

LED Light–emitting diode

MATLAB Matrix Laboratory

PLL Phase–Locked Loop

PWM Pulse Width Modulation

RAM Random Access Memory

ROM Read–Only Memory

SoC System on Chip

STP Shielded Twistted Pair

TTL Transistor–Transistor–Logic

UART Universal Asynchronous Receiver/Transmitter

USB Universal Serial Bus

VHDL VHSIC Hardware Description Language

VHSIC Very High Speed Integrated Circuits

Příloha H

Obsah přiloženého CD

```
|   ReadMe.txt - Soubor s popisem adresářové struktury CD
|
+---binary
|   +---fpga_system - Přeložený řídící systém připravený pro nahrání do FPGA
|   |       motor_driver.bit
|   |       motor_driver.mcs
|   \---pila_app - Obslužná aplikace pro řízení činnosti motoru
|           motor_p1.exe
|           pthreadGC2.dll
+---docs
|   |   bartimat-MI-Zadani_prace.pdf - Oficiální zadání diplomové práce
|   |
|   +---dc_dc_mp1584 - dokumentace k čipu MP1584
|   +---esmc-c2 - Dokumentace k lineárnímu motoru a jeho řídící jednotce
|   +---S3E_sample_pack - Dokumentace k vývojové desce s čipem Spartan-3E
|   +---saltek - Dokumentace ke kombinované přepěťové ochraně s VF filtrem
|   +---spartan-3_dev_board - Dokumentace k vývojové desce s čipem Spartan-3
|   \---xilinx_datasheets - Obecná dokumentace k čipům FPGA firmy Xilinx
+---pcb - Jednotlivé revize desek plošných spojů s výrobními podklady
|   +---rev1
|   +---rev2
|   \---rev3
+---sources - Adresář se zdrojovými kódy
|   +---fpga_system - Projektový adresář řídícího systému pro FPGA
|   \---pila_app - Projektový adresář obslužné aplikace
+---thesis - Text diplomové práce
|   +--- latex - Zdrojové kódy práce v jazyce LaTex
|   \---pdf - Výsledný PDF soubor s textem diplomové práce
\---video
        motor_run.mp4 - Demonstrační video z činnosti motoru
```