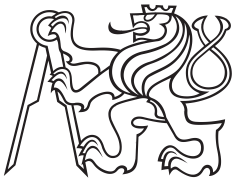
Bakalářská práce



České
vysoké
učení technické
v Praze

F2

Fakulta strojní
Ústav mechaniky, biomechaniky a mechatroniky

Vektorové řízení PMSM motoru Yaskawa SGMM-A1S312 na platformě LaunchPad F28069M v prostředí Simulink/Matlab

Tomáš Buriánek

Vedoucí: Ing. Martin Nečas, MSc., Ph.D.
Studijní program: Teoretický základ strojního inženýrství
Srpen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Buriánek** Jméno: **Tomáš** Osobní číslo: **482391**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav mechaniky, biomechaniky a mechatroniky**

Studijní program: **Teoretický základ strojního inženýrství**

Studijní obor: **bez oboru**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Vektorové řízení PMSM motoru Yaskawa SGMM-A1S312 na platformě LaunchPad F28069M v prostředí Simulink/Matlab

Název bakalářské práce anglicky:

Vector control of PMSM Yaskawa SGMM-A1S312 motor on LaunchPad F28069M platform in Simulink/Matlab environment

Pokyny pro vypracování:

1. Seznamte se s problematikou řízení PMSM motorů vektorovou metodou
2. Sestavte řídicí algoritmus motoru Yaskawa SGMM-A1S312 v prostředí Simulink
3. Proveďte naladění regulátorů řídicího algoritmu
4. Implementujte řídicí algoritmus na platformě LaunchPad F28069M
5. Kriticky zhodnoťte dosažené výsledky

Seznam doporučené literatury:

Literatura:
[1] UHLÍŘ, Ivan. Elektrické stroje a pohony. Vyd. 2., přeprac. Praha: Nakladatelství ČVUT, 2007. ISBN 978-80-01-03730-0.
[2] ELLIS, George. Control system design guide: using your computer to understand and diagnose feedback controllers. Butterworth-Heinemann, 2012.
[3] PULLE, Duco WJ; DARNELL, Pete; VELTMAN, André. Applied control of electrical drives. Springer, New York, 2015.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Martin Nečas, MSc., Ph.D., odbor mechaniky a mechatroniky FS

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **28.04.2021** Termín odevzdání bakalářské práce: **13.08.2021**

Platnost zadání bakalářské práce: _____

Ing. Martin Nečas, MSc., Ph.D. podpis vedoucí(ho) práce	doc. Ing. Miroslav Španiel, CSc. podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ _____

Datum převzetí zadání Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Poděkování

Chtěl bych tímto poděkovat vedoucímu práce za poskytnutí motoru a použitého hardwaru a za odbornou pomoc při tvorbě této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

V Kolíně, 11. srpna 2021

Abstrakt

Tato práce je zaměřena na řízení PMSM motoru Yaskawa SGMM-A1S312 vektorovou metodou. Řízení bylo realizováno pomocí vývojové desky LaunchPad F28069M s výkonovým modulem BOOSTXL-DRV8301. V první části práce je popsán princip vektorového řízení, princip realizace vypočítaného akčního zásahu pomocí vektorové modulace a návrh parametrů regulátorů na základě matematického modelu. Ve druhé části práce je pak popsána realizace uvedených principů při řízení skutečného motoru. Řídicí algoritmus je navržen pomocí blokových schémat v programu Simulink. Řízení bylo zprovozněno na úrovni polohové regulace.

Klíčová slova: řízení motoru, PMSM, vektorová metoda, FOC, vektorová modulace, SVM, regulace, regulátory, Simulink, mikrokontrolér

Vedoucí: Ing. Martin Nečas, MSc., Ph.D.

Abstract

This thesis is focused on the vector control of the PMSM engine Yaskawa SGMM-A1S312. The vector control was realized using the LaunchPad F28069M development kit with the BOOSTXL-DRV8301 BoosterPack. The first part of the thesis describes the principle of vector control, the principle of implementation of the control action intervention using space vector modulation and the tuning of PI controller parameters based on a mathematical model. The second part of the thesis then describes the implementation of these principles in the control of a real engine. The control algorithm is designed using block diagrams in Simulink. The vector control was managed at a level of position regulation.

Keywords: motor control, PMSM, vector control, field-oriented control, FOC, space vector modulation, SVM, regulation, regulator, controller, Simulink, microcontroller

Obsah

1 Úvod 1

Část I
Teorie vektorového řízení PMSM motoru

2 Princip FOC 5
2.1 Konstrukce vektoru proudu 5
2.2 Tvorba krouticího momentu 6
2.3 Řízení krouticího momentu 6
2.4 Clarke-Parkova transformace 8
2.4.1 Clarkeové transformace 8
2.4.2 Parkova transformace 9
2.5 Regulace 9
2.6 Zpětná Clarke-Parkova transformace 11
2.6.1 Zpětná Parkova transformace 11
2.6.2 Zpětná Clarkeové transformace 11
2.7 Vícepólové motory 12

3 Vektorová modulace 13
3.1 Princip SVM 13
3.2 Výpočet pomocí upravené zpětné Clarkeové transformace 15

4 Regulace 19
4.1 Kaskádová regulace 19
4.2 Druhy PI regulátorů 19
4.3 Návrh proudových regulátorů . . 20
4.4 Návrh rychlostních regulátorů . . 22
4.5 Anti-windup 23

Část II
Praktická realizace vektorového řízení

5 Použité přístroje, zařízení a software 27
5.1 Přístroje a zařízení 27
5.2 Software 29

6 Základní koncepce a hardwarové řešení 31
6.1 ePWM modul 31
6.2 Měření proudů ADC převodníkem 34
6.2.1 Princip měření proudů 34
6.2.2 Konstrukce výkonového modulu 35
6.2.3 ADC modul 36
6.3 eQEP modul 38

7 Softwarové řešení řídicího algoritmu 41
7.1 Základní informace 41
7.1.1 Datové typy 41
7.1.2 Optimalizace kódu 42
7.1.3 Úlohy běžící na různých frekvencích 42
7.2 Inicializační rutina a ovládání motoru 43
7.2.1 SPI komunikace 44
7.3 Regulační smyčky 46
7.3.1 Proudová regulační smyčka . . 46
7.3.2 Rychlostní regulační smyčka . 47
7.3.3 Polohová regulační smyčka . . 49

8 Parametry regulátorů 51
8.1 Proudové regulátory 51
8.2 Rychlostní regulátor 52
8.3 Polohový regulátor 54

9 Doplňující informace a zhodnocení 57
9.1 Sériová komunikace 57
9.2 Omezení vyplývající z použití Simulinku 58

10 Závěr 59

Přílohy

A Schémata modelu 63

B Literatura 73

Obrázky

2.1 Konstrukce vektoru proudu 5
2.2 Souřadné systémy 6
2.3 Krouticí moment 7
2.4 Clarkeové transformace 9
2.5 Parkova transformace 10
2.6 Schéma proudové regulace 10
2.7 Zpětná Parkova transformace .. 11
2.8 Schéma FOC 12

3.1 Zapojení motoru pomocí tří půlmůstků 14
3.2 Vektory napětí SVM 14
3.3 SVM pomocí střídavé reverzní sekvence 15
3.4 Porovnání nastavení PWM 17
3.5 Průběhy napětí 18

4.1 Schéma kaskádové regulace 19
4.2 Sériový regulátor s anti-windup . 24

5.1 Vývojová deska LaunchPad F28069M [20] 28
5.2 Výkonový modul BOOSTXL-DRV8301 [25] 28
5.3 Pohled na použitý hardware 30
5.4 Pohled na použitý hardware 30

6.1 Zapojení motoru 32
6.2 Použité nastavení EPWMxA ... 33
6.3 Možná nastavení mrtvé zóny ... 34
6.4 Časové schéma měření proudů .. 37
6.5 Zapojení zdroje měřeného napětí [10] 38
6.6 Průběh signálů enkodéru pro variantu *Quadrature-clock Mode* a určení směru 39

7.1 Použité časové kroky 43
7.2 Ovládání motoru 43
7.3 Schéma Stateflow 44
7.4 Nastavení výkonového modulu pomocí SPI komunikace 45
7.5 Škálování změřených proudů ... 46
7.6 Škálování úhlu θ 47
7.7 Škálování výstupu proudové regulace 47
7.8 Porovnání přechodových charakteristik spojité a diskrétní verze rychlostního filtru 48

8.1 Měření zpětného elektromotorického napětí 53
8.2 Porovnání přechodových charakteristik rychlostní regulace . 54
8.3 Přechodová charakteristika rychlostní regulace pro finální parametry 55
8.4 Polohová regulace při sinusovém průběhu žádané hodnoty 56

Tabulky

3.1 Vektory napětí SVM 14
3.2 Určení sektoru 16
3.3 Výpočet T_1 a T_2 16
3.4 Výpočet hodnot pro registry PWM modulů 17

5.1 Parametry motoru 27

6.1 Maximální měřitelné proudy a proudová rozlišení v závislosti na použitém zesílení napětí 36

Kapitola 1

Úvod

Elektrické motory jsou každodenní součástí našeho života. Setkáváme se s nimi téměř všude, od domácích spotřebičů, přes elektromobily až po průmyslové stroje. Různé aplikace vyžadují různé typy elektromotorů s různě pokročilými metodami řízení. Možnosti řízení se výrazně rozšířily s příchodem mikrokontrolérů, které umožňují digitálně provádět různé výpočty, vyhodnocovat data ze senzorů a zároveň promítnout vypočítané požadavky zpět do řízených objektů.

Jednou z metod řízení použitelných pro trojfázové AC motory a bezkartáčové DC motory je vektorová metoda (*Vector control*), běžně nazývaná též *Field-oriented control* (FOC). Je založena na souboru matematických operací zjednodušujících regulaci proudů v motoru, které přímo souvisejí s krouticím momentem motoru a celkově tak i s jeho pohybem. Její postup se mírně liší v závislosti na použitém typu motoru, v této práci se budeme zabývat jejím využitím na trojfázových synchronních motorech s permanentními magnety (PMSM). Ty se běžně využívají v oblasti menších výkonů, ať už při řízení rychlosti nebo jako servomotory při řízení polohy.

Důležitou roli při řízení motorů FOC metodou hraje samotný mikrokontrolér a jeho možnosti. V druhé části práce je proto popsána realizace řízení PMSM motoru Yaskawa SGMM-A1S312 pomocí vývojové desky LaunchPad F28069M a výkonového modulu BOOSTXL-DRV8301. Návrh řídicího algoritmu je realizován v programu Simulink.

Chování motoru v konečném důsledku závisí na nastavení použitých regulátorů. V této práci se proto budeme zabývat také jejich naladěním. Vliv parametrů jednotlivých regulátorů je možné do určité míry odhadnout použitím matematického modelu a jeho analýzou.

Část I

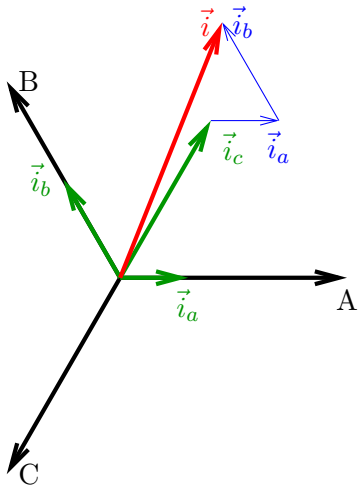
Teorie vektorového řízení PMSM motoru

Kapitola 2
Princip FOC

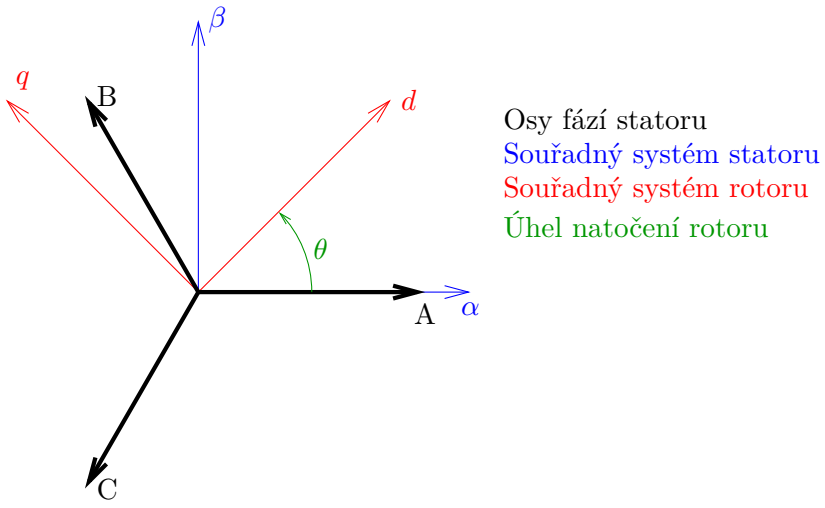
Základem řízení polohy každého servomotoru je řízení krouticího momentu. Proto je vhodné nejdříve popsat mechanismus jeho vzniku. Uvažujme nyní dvoupólový motor s permanentními magnety. Při provozu motoru vzniká točivé magnetické pole generované cívkami statoru. Na to reaguje magnetické pole rotoru tvořené permanentními magnety. Pro popis distribuce proudu v jednotlivých vinutích statoru a tím i velikosti a orientace magnetického pole statoru zavádíme pomyslný vektor proudu $\vec{i}$ (anglicky *space vector*). Magnetické pole rotoru zastupuje vektor magnetického indukčního toku $\vec{\psi}$. [1] [2] [3] [4]

2.1 Konstrukce vektoru proudu

Nejdříve vytvoříme schéma se třemi osami vzájemně pootočenými o 120° zastupujícími jednotlivá vinutí statoru. Proudy v jednotlivých vinutích znázorníme vektory $\vec{i}_a$, $\vec{i}_b$ a $\vec{i}_c$ ve směru příslušných os a o aktuální velikosti protékajícího proudu s respektováním směru. Výsledný vektor proudu reprezentující magnetické pole statoru získáme vektorovým součtem vektorů proudů jednotlivých fází, viz obr. 2.1. [4]



Obrázek 2.1: Konstrukce vektoru proudu



Obrázek 2.2: Souřadné systémy

2.2 Tvorba krouticího momentu

Krouticí moment můžeme nyní vyjádřit vztahem

$$\vec{M}_k = \vec{\psi} \times \vec{i} \tag{2.1}$$

a jeho velikost

$$M_k = \psi \, i \, \sin\delta \tag{2.2}$$

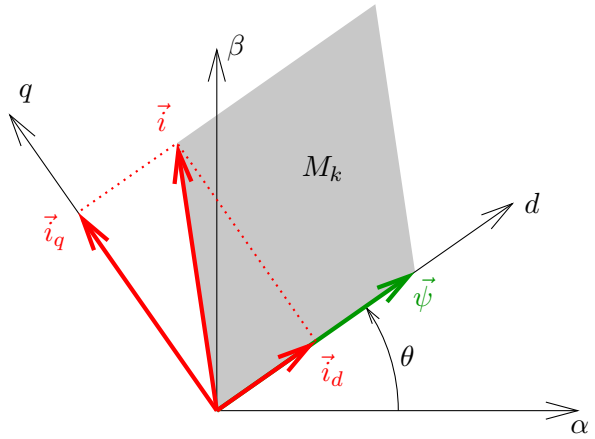
kde δ je úhel, který spolu svírají vektor proudu $\vec{i}$ s vektorem magnetického indukčního toku $\vec{\psi}$. Z průběhu funkce sinus je zřejmé, že maximálního momentu bude dosaženo při úhlu $\delta = 90°$. Proud $\vec{i}$ lze také rozložit do dvou na sebe kolmých složek $\vec{i}_d$ a $\vec{i}_q$, kde vektor $\vec{i}_d$ má směr shodný s vektorem magnetického indukčního toku $\vec{\psi}$ a vektor $\vec{i}_q$ je na něj kolmý. V takovém případě tvoří krouticí moment pouze kolmá, tzv. momentotvorná složka $\vec{i}_q$ a rovnici 2.2 lze přepsat do tvaru

$$M_k = \psi \, i_q \tag{2.3}$$

Složka $\vec{i}_d$ se nazývá tokotvorná. V případě indukčních motorů zajišťuje vznik magnetického indukčního toku. U PMSM motorů je tok $\vec{\psi}$ tvořen permanentními magnety umístěnými na rotoru. Velikost krouticího momentu je graficky znázorněna na obrázku 2.3, kde odpovídá velikosti šedé oblasti. [1] [2] [3] [4]

2.3 Řízení krouticího momentu

Považujme nyní magnetický indukční tok rotoru za konstantní. Z rovnice 2.3 vyplývá, že velikost krouticího momentu závisí pouze na velikosti momentotvorné složky proudu. Při řízení motoru se tedy snažíme regulovat proudy v jednotlivých fázích tak, aby výsledný vektor proudu byl kolmý na vektor magnetického indukčního toku a jeho velikost zajistila požadovaný krouticí moment. Postup můžeme rozdělit do následujících čtyř bodů:



Obrázek 2.3: Krouticí moment

1. Změření proudů protékajících jednotlivými vinutími motoru.
2. Porovnání změřených proudů s požadovanými a výpočet regulačních odchylek.
3. Vytvoření korekčních napětí zesílením regulačních odchylek.
4. Upravení napětí na svorkách motoru.

Měření proudů bude podrobně popsáno v kapitole 6.2, nyní předpokládejme, že je máme změřené. Je však vhodné zmínit, že motor se chová jako elektrický uzel. Díky tomu stačí změřit proudy ve dvou libovolných fázích, neboť proud tekoucí poslední fází lze dopočítat z prvního Kirchhoffova zákona.

Jak již bylo zmíněno, velikost krouticího momentu závisí na složce vektoru proudu kolmé na vektor magnetického indukčního toku. Ten je pevně spojen s rotorem, nutně tedy musíme znát také úhel natočení rotoru θ, který nám zároveň určuje natočení rotorového souřadného systému, viz obr. 2.2. Úhel θ volíme od kladné osy fáze A k vektoru magnetického indukčního toku $\vec{\psi}$. Tento krok nám později usnadní transformaci vektorů mezi souřadnými systémy. Kladný směr tohoto úhlu volíme proti směru otáčení hodinových ručiček. Kladný směr otáčení rotoru je s ním shodný. Určení úhlu natočení rotoru je možné provést pomocí senzoru nebo bezsenzorovou metodou, nyní však předpokládejme, že jej známe.

Abychom mohli změřené hodnoty proudů porovnat s požadovanými, musíme je transformovat do stejného souřadnicového systému. Z principu máme dvě možnosti – transformovat žádaný vektor proudu $\vec{i}^{ref}$ do nehybného souřadného systému statoru a rozdělit ho mezi jednotlivé fáze nebo transformovat změřený vektor proudu $\vec{i}$ do rotujícího souřadného systému rotoru. Pokud bychom však zvolili první variantu, museli bychom, za předpokladu konstantních otáček a konstantního zatížení, regulovat sinusové průběhy. Při zvolení druhé varianty budeme za stejných předpokladů regulovat konstantní hodnoty. Při řízení motorů metodou FOC se proto běžně využívá varianta druhá. Její realizaci provedeme pomocí tzv. Clarke-Parkovy transformace. Souřadné systémy jsou společně s jejich označením znázorněny na obrázku 2.2. [1] [3] [4]

2.4 Clarke-Parkova transformace

2.4.1 Clarkeové transformace

Prvním krokem po změření proudů jednotlivých fází je určení výsledného vektoru. To můžeme provést rozkladem vektorů jednotlivých fází pomocí trigonometrických funkcí soustavou rovnic

$$\begin{bmatrix} i_\alpha \\ i_\beta \\ 0 \end{bmatrix} = \begin{bmatrix} \cos 0° & \cos 120° & \cos 240° \\ \sin 0° & \sin 120° & \sin 240° \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{2.4}$$

kde i_α a i_β jsou složky výsledného vektoru $\vec{i}$ ve směru příslušných os statorového souřadného systému. Poslední rovnice této soustavy zastupuje již zmíněný Kirchhoffův zákon. Při řízení motorů FOC metodou se však běžně využívá Clarkeové transformace ve standardním amplitudově invariantním tvaru

$$\begin{bmatrix} i_\alpha \\ i_\beta \\ 0 \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{2.5}$$

kde jsou trigonometrické funkce vyčísleny a poslední rovnice je rozšířena $\frac{1}{2}$. Hlavním rozdílem je zde však přidání škálovacího součinitele $\frac{2}{3}$. Běžnou praxí je totiž naškálovat změřené hodnoty proudů i_a, i_b a i_c tak, aby maximálnímu proudu odpovídala hodnota 1. Amplituda výsledného vektoru je však potom větší. Přidaný součinitel v transformaci zmenší výsledný vektor tak, aby složky i_α a i_β, stejně tak jako výsledný vektor $\vec{i}$, dosahovali maxima opět na hodnotě 1. Jako kompenzace tohoto kroku se někdy zavádí škálovací součinitel $\frac{3}{2}$ do rovnice 2.1.

Úpravou soustavy rovnic 2.5 získáme rovnice

$$i_\alpha = i_a \tag{2.6a}$$

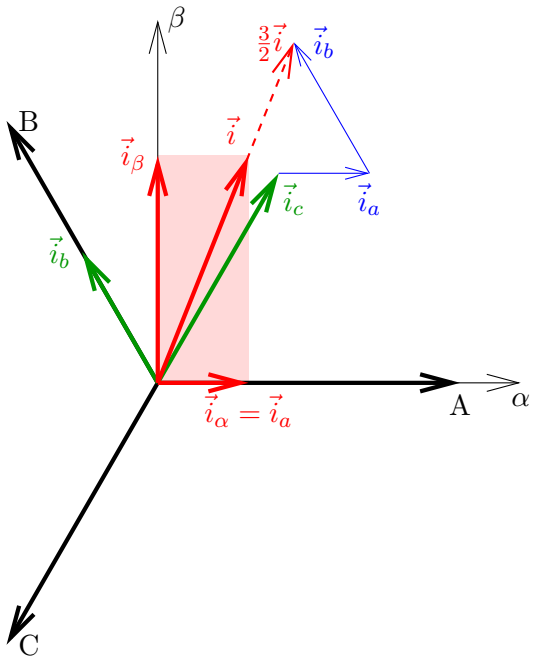
$$i_\beta = \frac{i_a + 2\,i_b}{\sqrt{3}} \tag{2.6b}$$

$$0 = i_a + i_b + i_c \tag{2.6c}$$

které se používají při samotném výpočtu. Clarkeové transformace je znázorněna na obrázku 2.4.

Převedení vektorů z os jednotlivých fází nám poskytuje významnou výhodu. Původně jsme měli vektor proudu popsaný pomocí tří složek, což je vzhledem k jeho dimenzi zbytečně moc. Nyní k jeho popisu používáme složky pouze dvě. Tím jsme redukovali potřebný počet regulátorů ze tří na dva.

Dále si můžeme povšimnout, že rovnice 2.6a a 2.6b reprezentující výpočet velikosti složek i_α a i_β neobsahují velikost proudu ve fázi C. Pokud tedy měříme proudy vždy ve fázích A a B, nemusíme proud i_c vůbec dopočítávat a rovnici 2.6c už nepotřebujeme (je obsažena v předchozích dvou rovnicích). [4] [5] [6] [7]



Obrázek 2.4: Clarkeové transformace

2.4.2 Parkova transformace

Máme-li vektor proudu $\vec{i}$ popsaný pomocí složek $\vec{i}_\alpha$ a $\vec{i}_\beta$ ve směrech os statorového souřadného systému a známe-li úhel natočení rotoru θ, můžeme přejít k dalšímu kroku. Tím je tzv. Parkova transformace popsaná rovnicemi

$$i_d = \quad i_\alpha \cos\theta + i_\beta \sin\theta \tag{2.7a}$$

$$i_q = -i_\alpha \sin\theta + i_\beta \cos\theta \tag{2.7b}$$

kde i_d a i_q jsou složky výsledného vektoru $\vec{i}$ ve směru rotorového souřadného systému. Rovnice je možné zapsat taktéž v maticovém tvaru

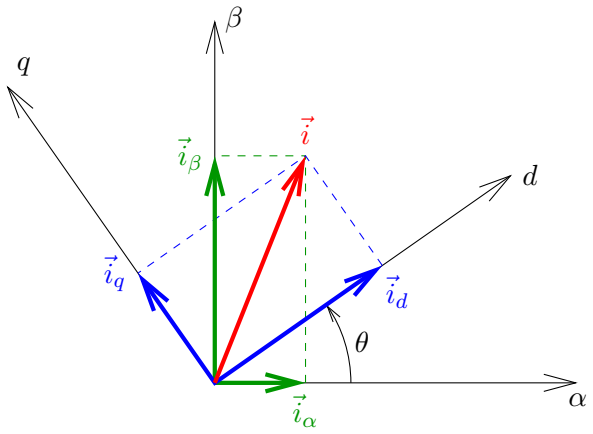
$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \tag{2.8}$$

ze kterého je patrné, že se jedná čistě o násobení maticí rotace. Parkova transformace je zobrazena na obrázku 2.5. [4] [6] [8]

2.5 Regulace

Stále se nacházíme ve druhém bodě výše uvedeného postupu. Podařilo se nám získat vektor proudu a převést ho do složek $\vec{i}_d$ a $\vec{i}_q$. Nyní je potřeba určit, jak mají vypadat požadované hodnoty těchto proudů a porovnat je s nimi.

Jak již bylo zmíněno, netvoří složka $\vec{i}_d$ žádný moment, ale vytváří magnetický indukční tok. V případě PMSM motorů však předpokládáme, že všechen potřebný



Obrázek 2.5: Parkova transformace

tok zajišťují permanentní magnety umístěné na rotoru. Regulací se tedy snažíme tuto složku eliminovat, a proto

$$\vec{i}_d^{\,ref} = 0 \tag{2.9}$$

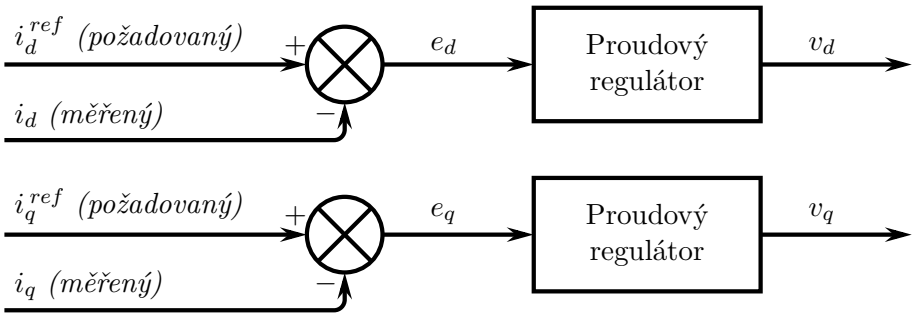
Naopak složka $\vec{i}_q$ je jediná, která krouticí moment tvoří, a velikost $\vec{i}_q^{\,ref}$ tedy přímo vychází z rovnice 2.3. Prakticky je tato rovnice zahrnuta v konstantách rychlostního regulátoru. Problematika regulátorů bude rozebrána v kapitole 4.

Známe-li tedy požadované velikosti složek, můžeme vytvořit regulační odchylky

$$e_d = i_d^{ref} - i_d \tag{2.10a}$$
$$e_q = i_q^{ref} - i_q \tag{2.10b}$$

jak je znázorněno na obrázku 2.6.

Dostáváme se ke třetímu bodu postupu. Zesílíme regulační odchylky pomocí proudových regulátorů, čímž získáme velikosti korekčních napětí $\vec{v}_d$ a $\vec{v}_q$ směřujících opět ve směrech os rotorového souřadného systému a reprezentujících výsledný vektor korekčního napětí $\vec{v}$. [4]



Obrázek 2.6: Schéma proudové regulace

2.6 Zpětná Clarke-Parkova transformace

2.6.1 Zpětná Parkova transformace

Poslední částí celého postupu je upravení napětí na svorkách motoru. Nejdříve je nutné vektor korekčního napětí převést zpět do statorového souřadného systému. K tomu slouží zpětná (inverzní) Parkova transformace popsaná rovnicemi

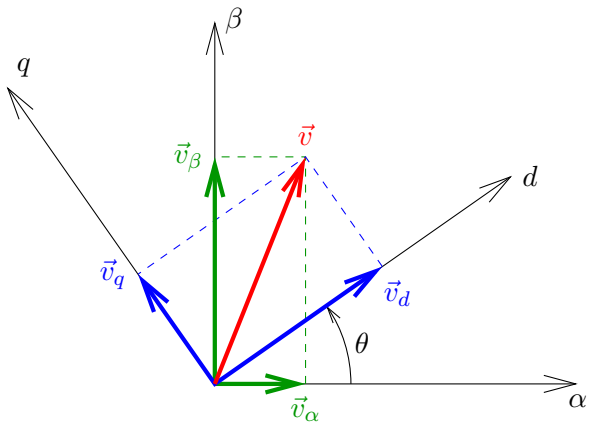
$$v_\alpha = v_d \cos\theta - v_q \sin\theta \tag{2.11a}$$

$$v_\beta = v_d \sin\theta + v_q \cos\theta \tag{2.11b}$$

kde v_α a v_β jsou velikosti vektorů směřujících ve směrech příslušných os statorového souřadného systému. Při přepsání do maticové podoby

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_d \\ v_q \end{bmatrix} \tag{2.12}$$

je patrné, že se nejedná o nic jiného než o násobení maticí inverzní k matici použité v Parkově transformaci dle rovnice 2.8. Zpětná Parkova transformace je znázorněna na obrázku 2.7. [4] [6]



Obrázek 2.7: Zpětná Parkova transformace

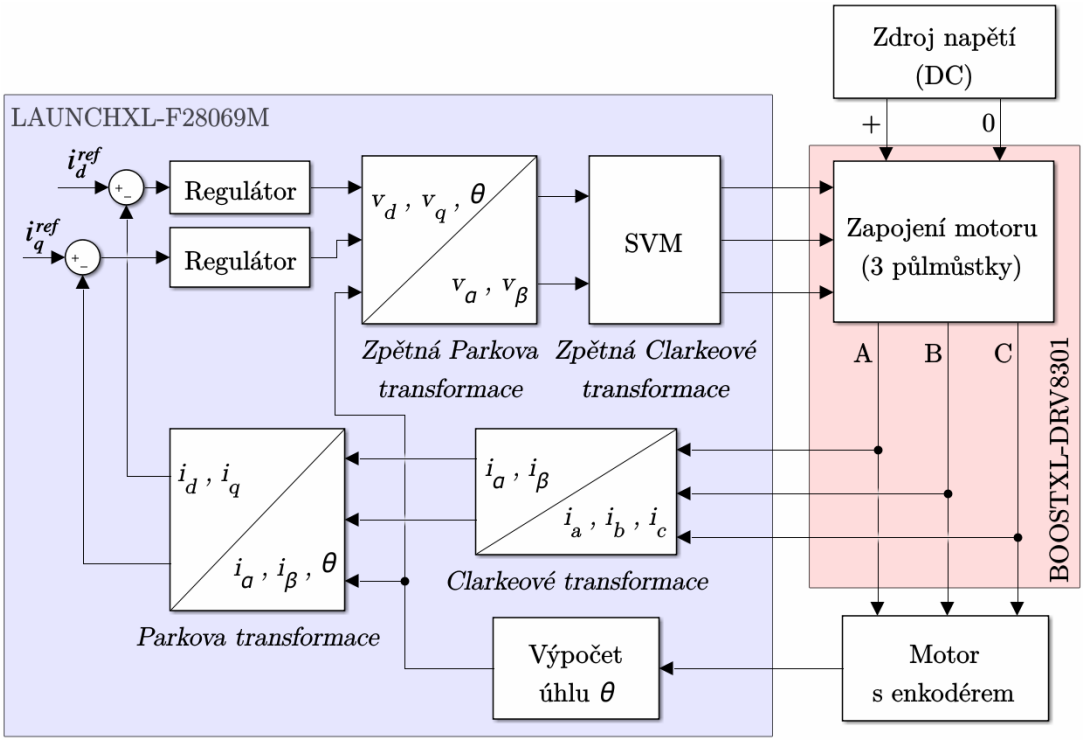
2.6.2 Zpětná Clarkeové transformace

Nakonec je nutné převést napětí ze statorového souřadného systému zpět na osy jednotlivých fází. Je zřejmé, že opět využijeme inverzi již použité transformace z převodu vektorů proudů, tentokrát zpětnou (inverzní) Clarkeové transformaci popsanou rovnicemi

$$v_A = v_\alpha \tag{2.13a}$$

$$v_B = -\frac{1}{2} v_\alpha + \frac{\sqrt{3}}{2} v_\beta \tag{2.13b}$$

$$v_B = -\frac{1}{2} v_\alpha - \frac{\sqrt{3}}{2} v_\beta \tag{2.13c}$$



Obrázek 2.8: Schéma FOC

Zpětná Clarkeové transformace je v modifikované podobě součástí výpočtů vektorové modulace, viz kapitola 3. Na obrázku 2.8 je zobrazeno schéma vektorového řízení. [4]

2.7 Vícepólové motory

Na začátku této kapitoly byl zaveden předpoklad dvoupólového motoru (s jedním pólpárem). V praxi se však můžeme běžně setkat i s vícepólovými motory. Popsaný princip vektorového řízení v takovém případě funguje téměř stejně. Jedinou výjimku tvoří úhel natočení rotoru θ a velikost krouticího momentu.

V předchozím textu jsme nerozlišovali mezi mechanickým a elektrickým úhlem, neboť u dvoupólového motoru jsou shodné. U vícepólových motorů je však nutné uvažovat úhel θ jako elektrický. Převod mechanického úhlu (obvykle získaného mechanickým senzorem, např. enkodérem) na elektrický můžeme provést vztahem

$$\theta_{elektrický} = p \cdot \theta_{mechanický} = \frac{P}{2} \cdot \theta_{mechanický} \tag{2.14}$$

kde p je počet pólpárů a P je počet pólů motoru.

Krouticí moment zavedený v kapitole 2.2 je také závislý na počtu pólů motoru. Pro vícepólové motory je proto nutné upravit rovnici 2.3 do tvaru

$$M_k = \frac{P}{2} \psi \, i_q = p \, \psi \, i_q \tag{2.15}$$

Stále platí, že takto zapsaná rovnice musí být dále škálována s ohledem na škálování změřených proudů a škálovací parametr Clarkeové transformace. [3]

Kapitola 3
Vektorová modulace

V minulé kapitole jsme stanovili velikost napětí na jednotlivých fázích motoru potřebnou pro správnou funkci vektorového řízení. Nyní je nutné daná napětí reálně vytvořit. Vzhledem k tomu, že výpočet potřebného napětí probíhá digitálně uvnitř mikroprocesoru, je nutné napětí alespoň zdánlivě převést na analogové hodnoty, aby mohlo být přivedeno na svorky motoru. K tomu máme dvě možnosti, buď použijeme analogově digitální převodník nebo pulzně šířkovou modulaci (zkráceně PWM, z anglického ***P**ulse **W**idth **M**odulation*). Dále se budeme zabývat pouze druhou variantou.

Princip PWM signálu spočívá ve velmi rychlém střídání maximálního a nulového napětí – logická 1 a 0. Řízený objekt vlivem setrvačnosti, respektive fázového posunutí, na tyto změny nestíhá reagovat a chová se, jako by byl napájen spojitým napětím odpovídajícím průměrné hodnotě modulovaného napětí.

Při vektorovém řízení se dnes běžně používá vektorová modulace (zkráceně SVM, z anglického ***S**pace **V**ector **M**odulation*). Jedná se o metodu, která umožňuje vhodně zvolit střídání logických stavů stejnosměrného napětí V_{DC} na jednotlivých svorkách motoru. [9]

3.1 Princip SVM

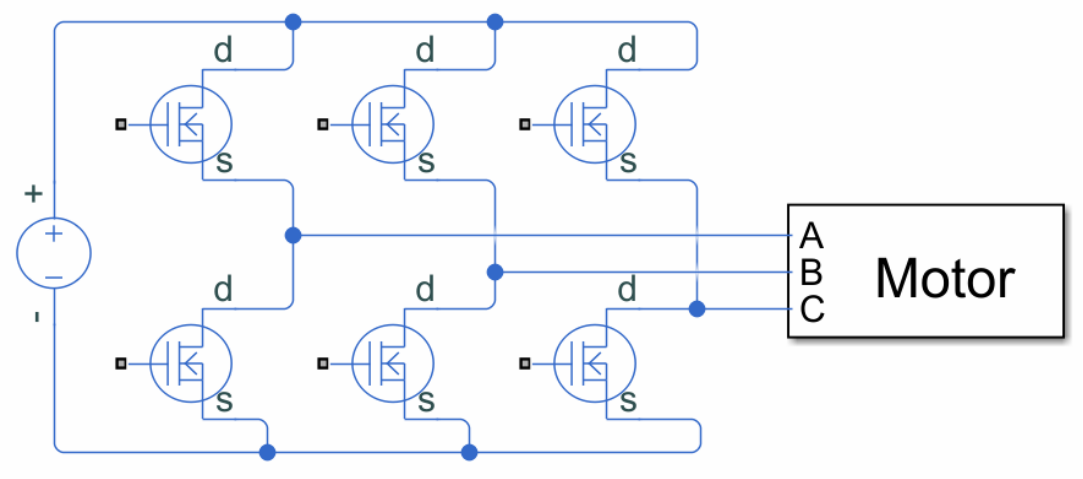
Předpokládejme zapojení motoru pomocí tří půlmůstků dle obrázku 3.1. Pomocí PWM signálů můžeme spínat jednotlivé tranzistory. Budeme-li dále předpokládat, že u každé fáze bude vždy jeden tranzistor sepnutý a druhý ne (fáze připojena na logickou 1 nebo 0), máme k dispozici osm možností, jak tranzistory spínat, viz tabulka 3.1. Šest z těchto možností představuje vektory napětí po 60°, zbylé dvě nulové vektory, viz obr. 3.2.

Střídáním dvou sousedních vektorů v kombinaci s nulovým vektorem během jedné spínací periody můžeme vytvořit libovolný vektor $\vec{v}^{\,ref}$. Pokud bychom chtěli vytvořit například vektor nacházející se v intervalu $\langle 120°, 180°\rangle$, střídali bychom vektory $\vec{v}_{120}$, $\vec{v}_{180}$ a $\vec{v}_{00}$. Úhel výsledného vektoru v takovém případě závisí na poměru časů T_1 a T_2 napěťových vektorů $\vec{v}_{120}$ a $\vec{v}_{180}$ a velikost na poměru doby trvání T_0 nulového vektoru $\vec{v}_{00}$ vůči délce spínací periody. Časové úseky je možné definovat pomocí vztahů

$$T_1 = T\, m\, \sin\left(60 - \gamma\right) \tag{3.1a}$$

$$T_2 = T\, m\, \sin\left(\gamma\right) \tag{3.1b}$$

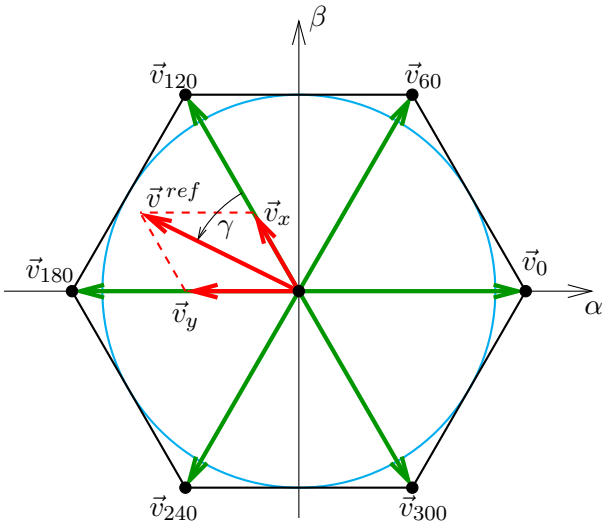
$$T_0 = T - T_1 - T_2 \tag{3.1c}$$



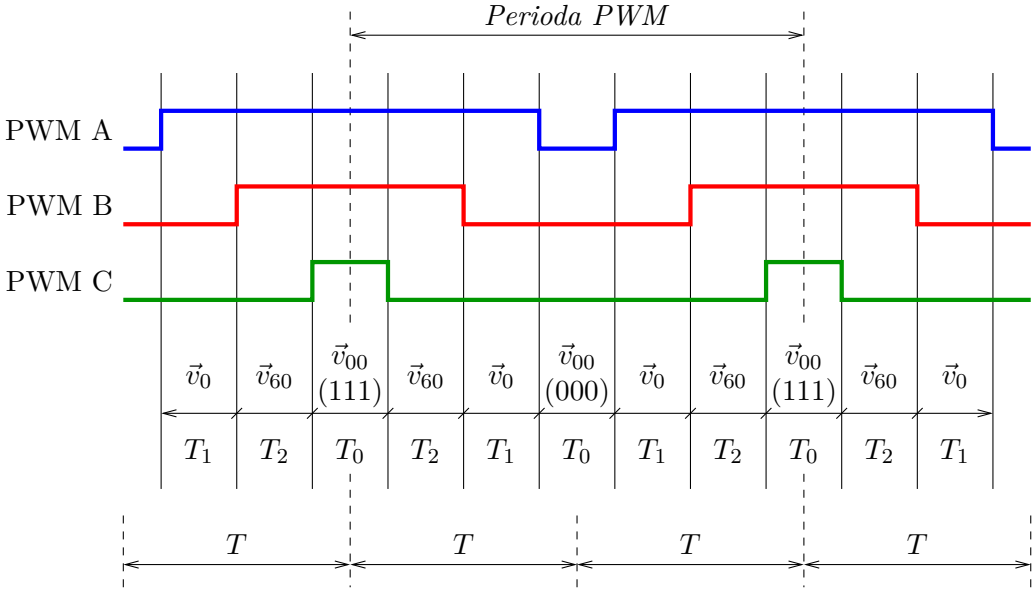
Obrázek 3.1: Zapojení motoru pomocí tří půlmůstků

Vektor	Fáze A	Fáze B	Fáze C	v_{ab}	v_{bc}	v_{ca}
$\vec{v}_{00}$	0	0	0	0	0	0
	1	1	1	0	0	0
$\vec{v}_0$	1	0	0	V_{DC}	0	-V_{DC}
$\vec{v}_{60}$	1	1	0	0	V_{DC}	-V_{DC}
$\vec{v}_{120}$	0	1	0	-V_{DC}	V_{DC}	0
$\vec{v}_{180}$	0	1	1	-V_{DC}	0	V_{DC}
$\vec{v}_{240}$	0	0	1	0	-V_{DC}	V_{DC}
$\vec{v}_{300}$	1	0	1	V_{DC}	-V_{DC}	0

Tabulka 3.1: Vektory napětí SVM



Obrázek 3.2: Vektory napětí SVM



Obrázek 3.3: SVM pomocí střídavé reverzní sekvence

kde T je spínací perioda, m amplituda napětí $\vec{v}^{ref}$ a γ je úhel měřený od vektoru napětí s nižším úhlem k referenčnímu napětí. T_1 je vždy doba trvání stavu odpovídajícího vektoru napětí s nižším úhlem $\vec{v}_x$ a T_2 doba trvání stavu odpovídajícího vektoru napětí s úhlem vyšším $\vec{v}_y$.

Existuje mnoho způsobů, jak řadit potřebné vektory v rámci jedné periody PWM tak, aby byly splněny požadované poměry časových úseků jednotlivých vektorů. V první řadě je totiž možné vytvořit nulový vektor pomocí dvou různých kombinací – $\vec{v}_{00}\,(000)$ a $\vec{v}_{00}\,(111)$, kde čísla v závorkách zastupují logické stavy fází A, B a C v uvedeném pořadí. Navíc je možné střídat vektory stále ve stejném pořadí nebo pomocí střídavé reverzní sekvence. Dále bude popsána pouze druhá varianta.

Technika střídavé reverzní sekvence využívá symetrické PWM. Její princip zachycuje obrázek 3.3. Je patrné, že dochází ke střídání vektorů $\vec{v}_{00}\,(000)$ a $\vec{v}_{00}\,(111)$ a že každá spínací perioda obsahuje oba tyto nulové vektory. Výhodou této metody je, že seřadí vektory tak, aby vždy došlo k přepnutí stavu pouze na jedné fázi, čímž snižuje počet jejich přepnutí. Další výhodou je, že v polovině periody PWM (případně na konci a na začátku – záleží na nastavení PWM modulů) jsou všechny fáze sepnuté na nulu. To umožňuje měření protékajících proudů pomocí měřicích odporů, viz kapitola 6.2. [1] [6] [9]

3.2 Výpočet pomocí upravené zpětné Clarkeové transformace

Výše uvedený způsob výpočtu časových úseků T_1, T_2 a T_0 předpokládá znalost amplitudy a úhlu vektoru $\vec{v}^{ref}$. To je vhodné při použití polárních souřadnic, při vektorovém řízení však máme vektor popsaný v kartézských souřadnicích. Výhodnější je proto výpočet pomocí upravené zpětné Clarkeové transformace.

Postup můžeme rozdělit do následujících čtyř bodů:

1. Provedení upravené zpětné Clarkeové transformace.
2. Určení sektoru, ve kterém se nachází vektor $\vec{v}^{ref}$.
3. Určení časových úseků T_1, T_2 a T_0.
4. Přímý výpočet hodnot pro registry PWM modulů.

Prvním krokem je výpočet upravené inverzní Clarkeové transformace popsané rovnicemi

$$i = \frac{\sqrt{3}}{2} v_\alpha - \frac{v_\beta}{2} \tag{3.2a}$$

$$j = v_\beta \tag{3.2b}$$

$$k = -\frac{\sqrt{3}}{2} v_\alpha - \frac{v_\beta}{2} \tag{3.2c}$$

Dále je potřeba určit sektor, ve kterém vektor $\vec{v}^{ref}$ leží. K tomu nám poslouží rovnice

$$N = sign\,(i) + 2sign\,(j) + 4sign\,(k) \tag{3.3}$$

kde

$$sign\,(+) = 1$$
$$sign\,(-) = 0$$

a tabulka 3.2, která určí sektor na základě hodnoty N z rovnice 3.3.

N	1	2	3	4	5	6
Sektor	6	2	1	4	5	3

Tabulka 3.2: Určení sektoru

Určení T_1, T_2 a T_0 provedeme pomocí tabulky 3.3 a rovnice

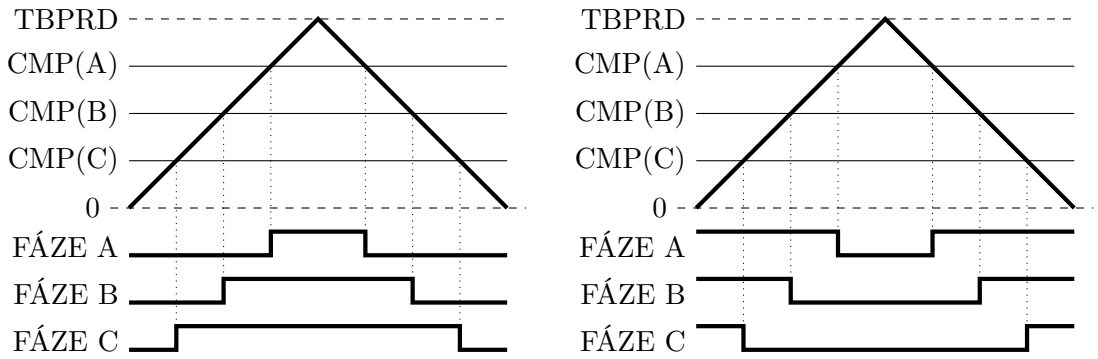
$$T_0 = 1 - T_1 - T_2 \tag{3.4}$$

Časové úseky jsou vztaženy na délku spínací periody T.

Sektor	1 $\langle 0°, 60° \rangle$	2 $\langle 60°, 120° \rangle$	3 $\langle 120°, 180° \rangle$	4 $\langle 180°, 240° \rangle$	5 $\langle 240°, 300° \rangle$	6 $\langle 300°, 360° \rangle$
T_1	i	$-k$	j	$-i$	k	$-j$
T_2	j	$-i$	k	$-j$	i	$-k$

Tabulka 3.3: Výpočet T_1 a T_2

Výpočet hodnot CMP pro registry jednotlivých PWM modulů (označení týkající se PWM modulů pro jednotnost převzato z dokumentace [10] k hardwaru použitému v praktické části této práce, viz kapitola 6.1) se odvíjí od použité metody SVM, nastavení PWM modulů a sektoru, ve kterém se požadovaný vektor napětí nachází.



Obrázek 3.4: Porovnání nastavení PWM

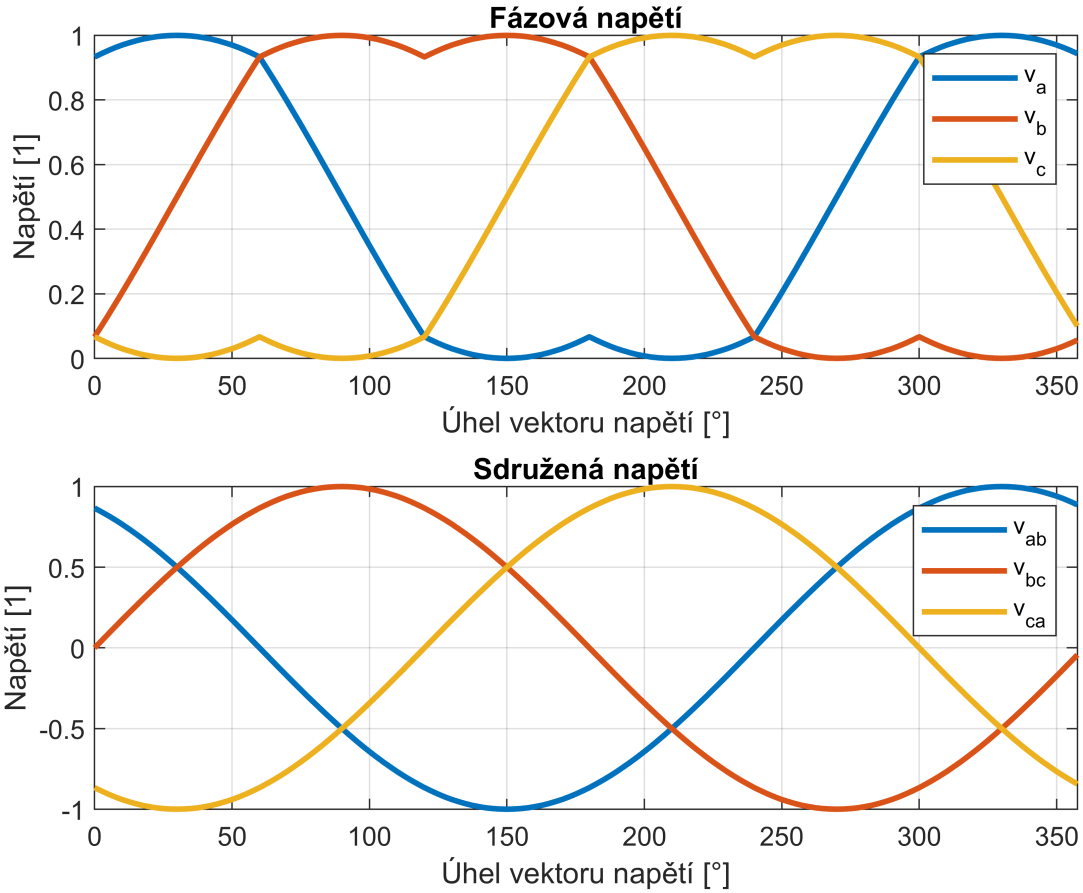
Sektor	1 $\langle 0°, 60° \rangle$	2 $\langle 60°, 120° \rangle$	3 $\langle 120°, 180° \rangle$
A (U)	$T_1 + T_2 + 0,5\,T_0$	$T_1 + 0,5\,T_0$	$0,5\,T_0$
B (V)	$T_2 + 0,5\,T_0$	$T_1 + T_2 + 0,5\,T_0$	$T_1 + T_2 + 0,5\,T_0$
C (W)	$0,5\,T_0$	$0,5\,T_0$	$T_2 + 0,5\,T_0$
Sektor	4 $\langle 180°, 240° \rangle$	5 $\langle 240°, 300° \rangle$	6 $\langle 300°, 360° \rangle$
A (U)	$0,5\,T_0$	$T_2 + 0,5\,T_0$	$T_1 + T_2 + 0,5\,T_0$
B (V)	$T_1 + 0,5\,T_0$	$0,5\,T_0$	$0,5\,T_0$
C (W)	$T_1 + T_2 + 0,5\,T_0$	$T_1 + T_2 + 0,5\,T_0$	$T_1 + 0,5\,T_0$

Tabulka 3.4: Výpočet hodnot pro registry PWM modulů

PWM moduly obsahují čítače, které po určitých časových intervalech (násobky periody cyklu procesoru) zvyšují hodnotu proměnné TBCTR. V případě symetrického PWM použitého při technice střídavé reverzní sekvence začne čítač po dosažení periody TBPRD hodnotu TBCTR opět snižovat, viz obr. 3.4. Pokaždé, když se CMP některé fáze rovná TBCTR, dojde k přepnutí logického stavu této fáze. V závislosti na tom, zda ke změně z logické 0 dané fáze na logickou 1 dojde při přičítání nebo odčítání, se mění průběh PWM signálu a tím i výpočet hodnoty CMP. Zde je vhodné připomenout, že logický stav fáze odpovídá logickému stavu PWM signálu řídícího horní tranzistor (připojuje fázi na napětí $+V_{DC}$), u dolního tranzistoru (připojuje fázi na nulu zdroje) je logický stav vždy opačný.

Pro případ zobrazený na obrázku 3.4 vpravo jsou výpočty pro jednotlivé fáze a sektory při použití techniky střídavé reverzní sekvence uvedeny v tabulce 3.4. V případě, že ke změně hodnoty TBCTR dochází po ± 1, stačí hodnoty vypočítané podle této tabulky vynásobit hodnotou TBPRD a získáme požadované hodnoty CMP, které zapíšeme do registrů jednotlivých PWM modulů.

Střídáním vektorů napětí nelze v motoru při většině úhlů vektoru referenčního napětí vytvořit výsledný vektor napětí s amplitudou rovnou napájecímu napětí. Vychází to z faktu, že součet délek dvou stran trojúhelníku je vždy menší nebo v extrémním případě



Obrázek 3.5: Průběhy napětí

roven délce třetí strany, pak ovšem trojúhelník degraduje v úsečku (je využit jen jeden z možných vektorů napětí). Oblast, v níž je možné výsledné vektory generovat, tvoří pravidelný šestiúhelník, viz obr. 3.2. Uvedený postup výpočtů však generuje vektory napětí pouze v rámci vepsané kružnice tohoto šestiúhelníku. Amplituda referenčního napětí je tedy výpočtem vždy zmenšena na $\sqrt{3}/2$ původní hodnoty.

Na obrázku 3.5 je zobrazen průběh fázových a sdružených napětí, která vznikají při použití popsaného postupu, v závislosti na úhlu vektoru napětí. Důležitý je zde hlavně průběh fázových napětí. Vzhledem k tomu, že jsou fázová napětí fázově posunuta o 120°, není nikdy zapotřebí maximálního napětí na více fázích současně. Proto můžeme k fázovým napětím přičíst další, společné napětí, které posouvá výchozí bod vektorů napětí jednotlivých fází. Tím můžeme zvýšit amplitudu sdružených napětí na $2/\sqrt{3}$ hodnoty, která by vznikla při sinusovém průběhu fázových napětí. To však nesouvisí se zmenšením amplitudy popsaným v předchozím odstavci a neeliminuje ho to. V případě použití sinusových průběhů fázového napětí místo vektorové modulace by došlo ke zmenšení amplitudy vektoru napětí na 3/4 původní hodnoty. Animaci vzniku uvedených fázových napětí je možné vidět ve videu [9]. [9]

Kapitola 4

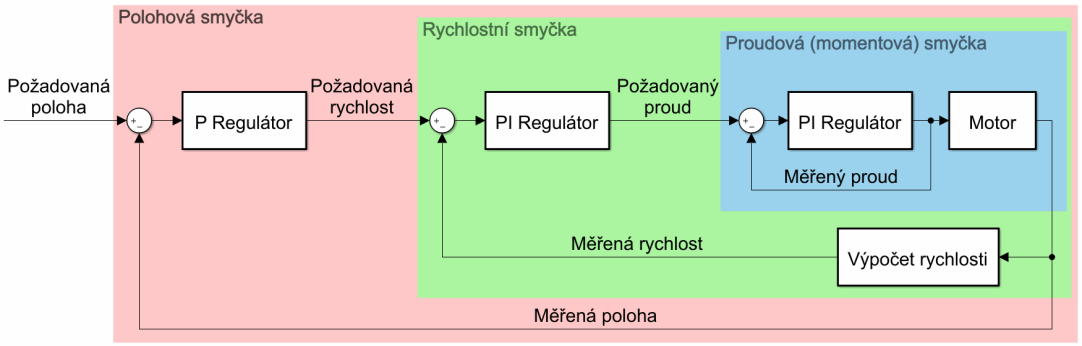
Regulace

V předchozích kapitolách jsme se zabývali principem vektorového řízení. Ten je víceméně univerzální. Velké rozdíly při jednotlivých aplikacích jsou naopak v parametrech regulátorů, které ovlivňují chování konkrétních systémů. Proto je jim věnována samostatná kapitola.

4.1 Kaskádová regulace

Při řízení servomotorů se dnes běžně využívá tzv. kaskádová regulace. Její princip spočívá v postupném nabalování uzavřených regulačních smyček na sebe, což usnadňuje ladění parametrů regulátorů. První smyčkou je proudová smyčka, která má přímou vazbu na krouticí moment motoru. Ta je součástí rychlostní smyčky a ta zase součástí smyčky polohové, jak je vidět na obrázku 4.1.

Nejprve naladíme parametry regulátoru proudové smyčky. Při ladění parametrů rychlostní smyčky již nemusíme zasahovat do smyčky proudové, neboť parametry regulátorů vnějších smyček neovlivní chování vnitřní smyčky. Stejně tak není ovlivněno chování proudové a rychlostní smyčky nastavením polohového regulátoru. [3]



Obrázek 4.1: Schéma kaskádové regulace

4.2 Druhy PI regulátorů

Při řízení motorů se dnes běžně využívají regulátory typu PI. Z hlediska uspořádání prvků uvnitř regulátoru můžeme rozdělit PI regulátory na dva druhy – sériový a

paralelní. PI regulátor v paralelní formě můžeme definovat přenosem

$$PI_{paralelní}(s) = \frac{K_i}{s} + K_p \tag{4.1}$$

kde K_p je proporcionální konstanta a K_i integrační konstanta. Výhodou paralelního regulátoru je, že přesně známe velikost proporcionální složky zajišťující zesílení aktuální regulační odchylky a velikost integrační složky sloužící k eliminaci trvalé regulační odchylky.

Nevýhodou však je chování jeho frekvenční charakteristiky v závislosti na jeho parametrech. Proporcionální konstanta má totiž vliv na zesílení regulátoru pouze ve vyšších frekvencích. V nízkých frekvencích je zesílení regulátoru dáno pouze integrační konstantou. Každá z těchto složek definuje ve frekvenční charakteristice jednu asymptotu. Asymptota pro frekvence jdoucí do nekonečna je tvořena konstantní funkcí závislou pouze na parametru K_p. Asymptota pro frekvence jdoucí k nule je tvořena klesající funkcí o sklonu 20 dB na dekádu závislou na parametru K_i. Poloha nuly přenosu určená průsečíkem těchto asymptot proto závisí na obou parametrech a je obtížné s ní při návrhu parametrů pracovat.

Provedeme-li substituci

$$K_a = K_p \tag{4.2a}$$

$$K_b = \frac{K_i}{K_p} \tag{4.2b}$$

můžeme definovat sériový regulátor s přenosem

$$PI_{sériový}(s) = \frac{K_a\,K_b}{s} + K_a = \frac{K_a\,K_b\left(1 + \frac{s}{K_b}\right)}{s} \tag{4.3}$$

V tomto případě má konstanta K_a vliv na zesílení regulátoru ve všech frekvencích a poloha nuly přenosu je určena pouze konstantou K_b. Toho lze využít při návrhu proudových regulátorů. [3] [11] [12] [13]

4.3 Návrh proudových regulátorů

Při návrhu parametrů proudového regulátoru je vhodné nejdříve zvolit vhodný matematický model motoru. K tomuto účelu využijeme aproximaci vinutí motoru v podobě sériového zapojení cívky, odporu a zdroje zpětného elektromotorického napětí. Vzhledem k tomu, že se zpětné elektromotorické napětí v závislosti na proudu obvykle mění pomalu, budeme ho považovat za konstantní. Nyní již můžeme napsat přenosovou funkci mezi napětím na svorkách motoru a vyvolaným proudem

$$\frac{I(s)}{V(s)} = \frac{\frac{1}{R}}{1 + \frac{L}{R}s} \tag{4.4}$$

Za předpokladu, že parametr K_a je škálován s ohledem na napájecí napětí V_{DC} a zmenšení vektoru napětí vektorovou modulací, můžeme napsat přenos zpětnovazební

smyčky

$$G(s) = \frac{PI(s)\,\dfrac{I(s)}{V(s)}}{1 + PI(s)\,\dfrac{I(s)}{V(s)}} \tag{4.5}$$

a po dosazení z rovnic 4.3 a 4.4 a úpravě dostaneme vztah

$$G(s) = \frac{1 + \dfrac{s}{K_b}}{\dfrac{L}{K_a\,K_b}s^2 + \left(\dfrac{R}{K_a\,K_b} + \dfrac{1}{K_b}\right)s + 1} \tag{4.6}$$

Vzhledem k tomu, že jmenovatel tohoto přenosu je druhého řádu, může být systém při nevhodném zvolení parametrů regulátoru kmitavý. Abychom tomu zabránili, položíme jmenovatele rovného součinu dvou mnohočlenů dle rovnice

$$\frac{L}{K_a\,K_b}s^2 + \left(\frac{R}{K_a\,K_b} + \frac{1}{K_b}\right)s + 1 = (1 + C\,s)(1 + D\,s) \tag{4.7}$$

kde C a D jsou reálná čísla. Po roznásobení pravé strany a porovnání výsledku s levou získáme podmínky

$$\frac{L}{K_a\,K_b} = C\,D \tag{4.8a}$$

$$\frac{R}{K_a\,K_b} + \frac{1}{K_b} = C + D \tag{4.8b}$$

Zvolíme-li nyní řešení rovnice 4.8b

$$\frac{R}{K_a\,K_b} = C \tag{4.9a}$$

$$\frac{1}{K_b} = D \tag{4.9b}$$

nahradíme jmenovatele v rovnici 4.6 dle rovnice 4.7 a provedeme substituci dle 4.9a a 4.9b, získáme pro zpětnovazební přenos vztah

$$G(s) = \frac{1 + \dfrac{s}{K_b}}{\left(1 + \dfrac{R}{K_a\,K_b}s\right)\left(1 + \dfrac{s}{K_b}\right)} = \frac{1}{1 + \dfrac{R}{K_a\,K_b}s} \tag{4.10}$$

Tím se nám podařilo systém s dvěma (potenciálně komplexními) póly a jednou nulou zjednodušit na systém s jedním reálným pólem a žádnou nulou. To má za následek úplnou eliminaci případných rezonancí či pouhých překmitů.

Dosadíme-li ještě z rovnic 4.9a a 4.9b zpět do rovnice 4.8a, získáme vztah

$$K_b = \frac{R}{L} \tag{4.11}$$

zajišťující splnění podmínek pro zjednodušení zpětnovazebního přenosu 4.6 do tvaru 4.10. Dosadíme-li nakonec podmínku 4.11 do vztahu 4.10, získáme finální vztah pro přenos

$$G(s) = \frac{1}{\dfrac{L}{K_a}s + 1} \tag{4.12}$$

odkud plyne, že

$$K_a = L \cdot \textit{šířka pásma} \tag{4.13}$$

Návrh se nám tak navíc zjednodušil na volbu jednoho parametru, který navíc přímo určuje šířku pásma. [3] [13]

4.4 Návrh rychlostních regulátorů

V případě rychlostní regulace opět použijeme sériový PI regulátor. Pro přehlednost však tentokrát označíme jeho konstanty K_c a K_d a rovnici 4.3 přepíšeme do tvaru

$$PI_{rychlostní}(s) = \frac{K_c\,K_d}{s} + K_c = \frac{K_c\,K_d\left(1 + \frac{s}{K_d}\right)}{s} \tag{4.14}$$

Při návrhu jeho parametrů opět vyjdeme z matematického modelu. Kromě již známého přenosu proudové regulační smyčky 4.12 a přenosu rychlostního regulátoru 4.14 musíme tentokrát zahrnout ještě vliv zátěže v podobě přenosu

$$Zátěž(s) = \frac{\omega(s)}{M_k(s)} = \frac{1}{J\,s + k_v} \tag{4.15}$$

kde J je moment setrvačnosti zátěže a k_v viskózní tlumení a vliv velikosti krouticího momentu v podobě přenosu mezi proudem i_q a momentem M_k

$$Mtr(s) = \frac{M_k(s)}{i_q(s)} = \frac{P}{2}\,\psi \tag{4.16}$$

Obvykle je také nutné filtrovat zpětnou vazbu v podobě vypočítané rychlosti, proto je nutné do matematického modelu zahrnout ještě přenos rychlostního filtru. Předpokládejme využití jednoduchého filtru v podobě přenosové funkce prvního řádu

$$Filtr(s) = \frac{1}{\tau\,s + 1} \tag{4.17}$$

kde τ je časová konstanta filtru.

Při zanedbání viskózního tlumení můžeme napsat přenosovou funkci otevřené smyčky

$$H(s) = \left(\frac{K_c\,K_d\left(1 + \frac{s}{K_d}\right)}{s}\right)\left(\frac{1}{\frac{L}{K_a}s + 1}\right)\left(\frac{P}{2}\,\psi\right)\left(\frac{1}{J\,s}\right)\left(\frac{1}{\tau\,s + 1}\right) \tag{4.18}$$

Nahradíme-li konstanty motoru a zátěže jednou konstantou

$$K = \frac{P\,\psi}{2\,J} \tag{4.19}$$

můžeme po zjednodušení rovnici 4.18 přepsat do tvaru

$$H(s) = \frac{K\,K_c\,K_d\left(1 + \frac{s}{K_d}\right)}{s^2\left(1 + \frac{L}{K_a}s\right)(1 + \tau\,s)} \tag{4.20}$$

Na základě analýzy této rovnice uvedené v [15] můžeme zavést tlumicí faktor δ a s jeho pomocí navrhnout konstanty rychlostního regulátoru na základě rovnic

$$K_d = \frac{1}{\delta^2 \tau} \tag{4.21a}$$

$$K_c = \frac{\delta K_d}{K} = \frac{1}{\delta K \tau} \tag{4.21b}$$

Čím vyšší hodnotu tlumicího faktoru zvolíme, tím menší bude překmit přechodové charakteristiky. Zároveň se ale prodlouží čas potřebný k dosažení ustálené hodnoty. Proto je doporučeno volit δ z rozsahu 2 až 35. [14] [15] [16]

Autor tohoto postupu ještě zavádí do rovnice 4.16 již zmíněný škálovací součinitel $\frac{3}{2}$. Pro přesnost je zde však vynechán, uvedený matematický model tak přímo reprezentuje reálný systém. Veškeré faktory ovlivňující škálování jsou místo toho vyhodnoceny současně, viz kapitola 8.2. Přesunutí škálovacího součinitele nemá na analýzu matematického modelu, naškálované parametry regulátoru ani na chování výsledného systému vliv. Změní se pouze nenaškálované parametry.

4.5 Anti-windup

Integrační složka regulátoru zajišťuje zohlednění minulosti regulačního procesu. Za normálních podmínek slouží k eliminaci trvalé regulační odchylky proporcionální složky. Pokud se však žádaná hodnota mění příliš rychle a velikost akčního zásahu dosáhne limitní hodnoty (dojde k saturaci), řízený objekt nezvládá reagovat na změnu žádané hodnoty a regulovaná veličina se mění pouze omezenou rychlostí. V takovém případě dochází k nežádoucí kumulaci integrační složky regulátoru vlivem velké regulační odchylky po celou dobu saturace řízeného objektu. V okamžiku, kdy regulovaná veličina dosáhne žádané hodnoty, způsobí akumulovaná hodnota integrační složky velmi velký překmit. Tomuto jevu se říká windup efekt.

V praxi k němu dochází například při rozběhu motoru, kdy je skokově změněna požadovaná rychlost. Vzhledem k velikosti překmitu je téměř vždy nutné tento efekt nějak omezit. Existuje více způsobů, jak toho dosáhnout. Jedním z nich je například omezení velikosti integrační složky na stejné limity, jaké má akční veličina. Tento způsob je velmi jednoduchý, nicméně jeho účinnost nemusí být dostatečná. V takovém případě je možné měnit limity integrační složky dynamicky podle velikosti proporcionální složky tak, aby součet proporcionální složky a příslušného limitu integrační složky (horního při kladné hodnotě proporcionální složky a dolního při záporné) byl roven příslušnému limitu akční veličiny.

Zavedeme-li označení limitů akční veličiny L (dolní limit) a H (horní limit), označení proporcionální složky P a označení limitů integrační složky I_{min} a I_{max}, můžeme statické limity integrátoru pro první uvedený případ definovat jako

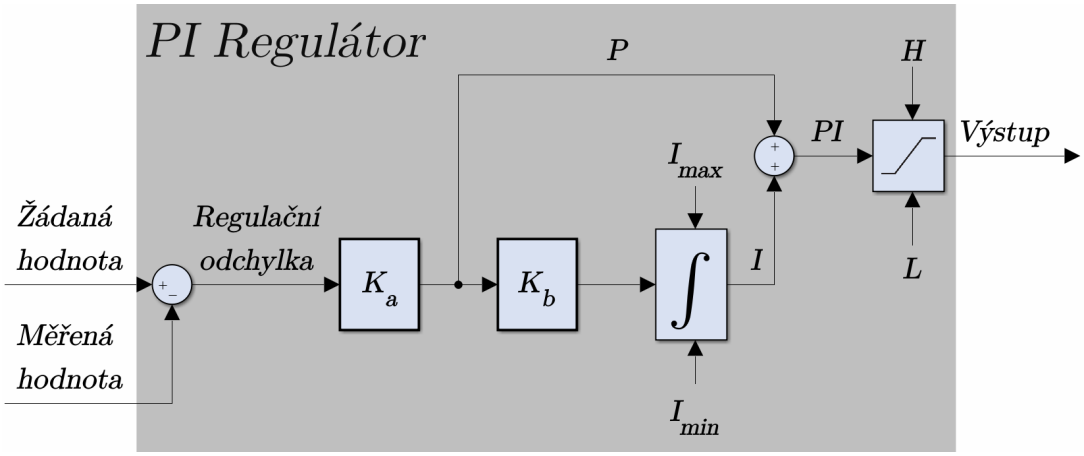
$$I_{max} = H \tag{4.22a}$$

$$I_{min} = L \tag{4.22b}$$

a dynamické limity pro druhý uvedený případ nastavit podmínkami

$$I_{max} = \max\left(H - P,\ 0\right) \tag{4.23a}$$

$$I_{min} = \min\left(L - P,\ 0\right) \tag{4.23b}$$



Obrázek 4.2: Sériový regulátor s anti-windup

Sériový PI regulátor s výše popsaným anti-windup opatřením je znázorněn na obrázku 4.2. [3] [17]

Část II

Praktická realizace vektorového řízení

Kapitola 5

Použité přístroje, zařízení a software

Cílem této práce bylo vytvoření řízení PMSM motoru Yaskawa SGMM-A1S312 pomocí vývojové desky LaunchPad F28069M a programu Simulink. To vyžaduje využití dalších přístrojů a zařízení a dalšího softwaru.

5.1 Přístroje a zařízení

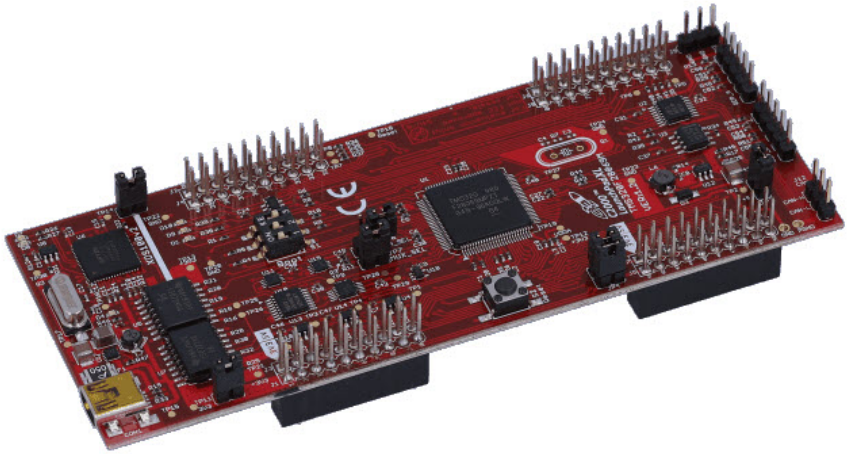
Motor Yaskawa SGMM-A1S312. Yaskawa SGMM-A1S312 je šestipólový trojfázový synchronní motor s permanentními magnety. Jeho základní parametry jsou uvedeny v tabulce 5.1. Motor obsahuje integrovaný inkrementální enkodér s 2048 pulzy na otáčku na každém kanálu (maximálního možného rozlišení 8192 poloh na otáčku lze dosáhnout při sledování náběžných i spádových hran na obou kanálech). [18]

Jmenovitý výkon	P_n	10 W
Jmenovité otáčky	n_n	3000 rpm
Maximální otáčky	n_{max}	5000 rpm
Jmenovitý proud	I_n	2,1 A_{rms}
Napájecí napětí	V_{DC}	24 V_{DC}
Moment setrvačnosti	J	$3,54 \cdot 10^{-7}$ kg $\cdot$ m^2
Odpor vinutí statoru[1]	R	0,9267 Ω
Indukčnost vinutí statoru[1]	L	$2,342 \cdot 10^{-4}$ H

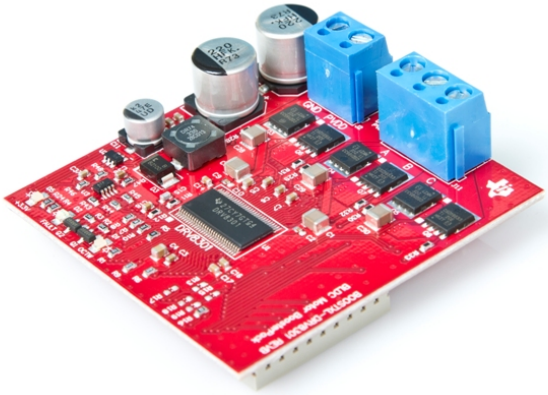
Tabulka 5.1: Parametry motoru

Vývojová deska LaunchPad F28069M. Pro řízení motoru byla použita vývojová deska LAUNCHXL-F28069M od společnosti Texas Instruments zobrazená na obrázku 5.1. Jejím základním prvkem je 32bitový procesor TMS320F28069M pracující na frekvenci 90 MHz s 256 kB paměti Flash a 96 kB paměti RAM. Deska disponuje šesti ePWM moduly (každý se dvěma kanály), sedmi 12bitovými ADC převodníky (opět každý se dvěma kanály), dvěma eQEP moduly pro zpracování signálu z inkrementálních enkodérů, SPI pro komunikaci s výkonovým modulem, USB pro komunikaci s počítačem a dalšími funkcemi. Deska pracuje hlavně s 3,3V logikou, obsahuje však i napájecí

[1] Označené veličiny výrobce neuvádí, jejich znalost je však důležitá při návrhu parametrů PI regulátorů. Proto byly převzaty z [19], kde byly experimentálně změřeny.



Obrázek 5.1: Vývojová deska LaunchPad F28069M [20]



Obrázek 5.2: Výkonový modul BOOSTXL-DRV8301 [25]

piny o napětí 5 V. S 5V logikou pracují také eQEP moduly. LaunchPad F28069M je navržen tak, aby bylo možné připojit až dva výkonové moduly a s jejich pomocí řídit až dva motory. [20] [21] [22] [23] [24]

Výkonový modul BOOSTXL-DRV8301. Jako výkonový modul k vývojové desce byl zvolen BOOSTXL-DRV8301, zobrazený na obrázku 5.2. Tento modul zajišťuje napájení motoru z externího zdroje stejnosměrného napětí pomocí tří půlmůstků tvořených MOSFET tranzistory, jak je znázorněno na obrázku 6.1. Základem modulu je ovladač půlmůstků DRV8301. Ovládání jednotlivých tranzistorů je prováděno PWM signály z hlavní desky LAUNCHXL-F28069M. Modul je dimenzován na napájecí napětí motoru 6–24 V a proud do 10 A RMS (14 A špička). Modul umožňuje měření proudů v jednotlivých fázích pomocí integrovaných měřicích odporů. [25] [26] [27] [28]

Zdroj napětí. Pro napájení byl použit laboratorní zdroj AX-3005DLS. Jedná se o stabilizovaný lineární zdroj stejnosměrného napětí, který umožňuje plynule regulovat napětí v rozsahu 0–30 V a nastavit omezení proudu v rozsahu 0–5 A. [29]

Osciloskop. Pro analýzu napětí na výstupech vývojové desky a na jednotlivých fázích motoru byl použit osciloskop OWON XDS3104AE. Jedná se o digitální 4kanálový osciloskop se 14bitovým rozlišením pracující na frekvenci 100 MHz. [30]

5.2 Software

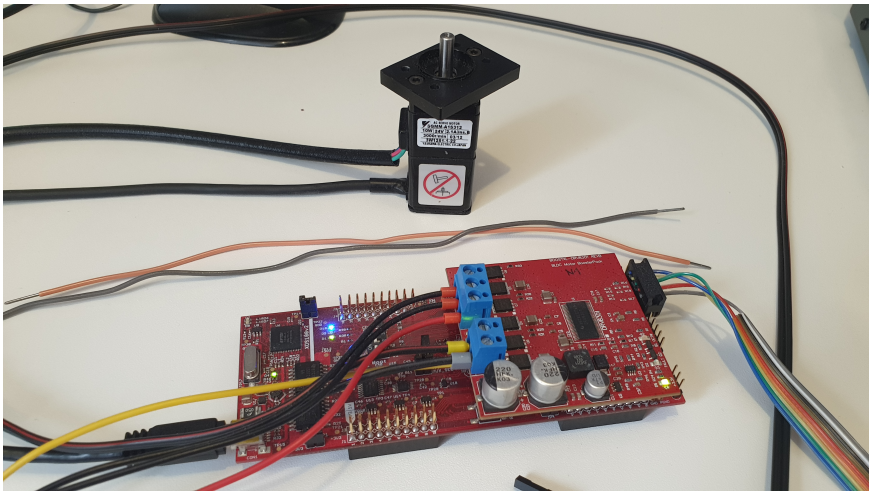
Software od Texas Instruments. Pro programování vývojových desek poskytuje společnost Texas Instruments (TI) specializovaný software. V této práci byly prostřednictvím balíčku hardwarové podpory v programu Simulink využity následující nástroje:

- **Code Composer Studio IDE (CCS).** CCS je integrované vývojové prostředí podporující mikrokontroléry a vestavěné procesory TI. Při práci byla použita verze 9.1.0. [31]
- **ControlSUITE.** ControlSUITE je sada softwarové infrastruktury, nástrojů a dokumentace navržená tak, aby pomohla zkrátit dobu vývoje. Při práci byla použita verze 3.4.9. [32]
- **C2000Ware.** C2000Ware je sada softwaru a dokumentace vytvořená za účelem minimalizace doby vývoje. Zahrnuje ovladače specifické pro zařízení, knihovny a příklady. Při práci byla použita verze 02.00.00.02. [33]

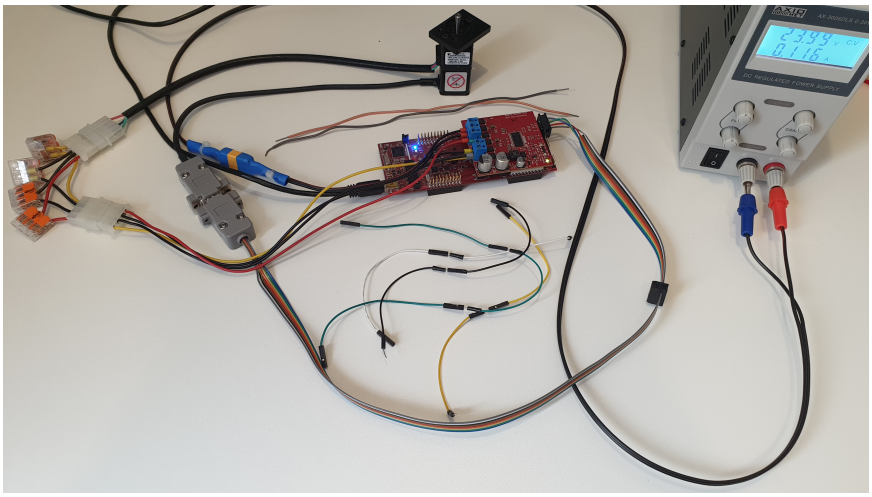
MATLAB, Simulink. Hlavním programem využitým v této práci je MATLAB, respektive jeho nadstavba Simulink. MATLAB je programovací platforma využívající jazyk MATLAB založený na maticových výpočtech. Obsahuje velké množství specializovaných toolboxů, které jeho oblast využití rozšiřují do téměř všech vědních a inženýrských oborů, kde se využívá matematický popis. [34]

Simulink je grafické programovací prostředí založené na práci s blokovými schématy. Je integrován do MATLABu a může využívat všechny jeho funkce. Umožňuje simulovat složité systémy a následně je i převést do C nebo C++ kódu prostřednictvím vhodných toolboxů. Pro práci s vývojovými deskami od TI je k dispozici balíček hardwarové podpory *Embedded Coder Support Package for Texas Instruments C2000 Processors*, který doplňuje do Simulinku knihovny bloků pro práci s jednotlivými moduly vývojových desek a pro generování optimalizovaného kódu určitých operací. [35] [36] [37]

Při práci byla využita verze programů MATLAB a Simulink R2020b.



Obrázek 5.3: Pohled na použitý hardware



Obrázek 5.4: Pohled na použitý hardware

Kapitola 6

Základní koncepce a hardwarové řešení

Základem celého řídicího algoritmu je proudová regulační smyčka. Ta je spouštěna pravidelně na frekvenci 20 kHz. Časovač PWM signálu vyvolá při dosažení periody zahájení konverze ADC převodníku. Ten po ukončení konverze vyvolá hardwarové přerušení, které spustí výpočet proudové regulační smyčky. Na začátku výpočtu proběhne nejdříve několik rozhodovacích operací (viz kapitola 7.2), které případně vynulují čítač polohy inkrementálního enkodéru, pak dojde k odečtení polohy. Následuje škálování změřených hodnot, vlastní regulační výpočet a zápis vypočítaných hodnot do registrů PWM modulů.

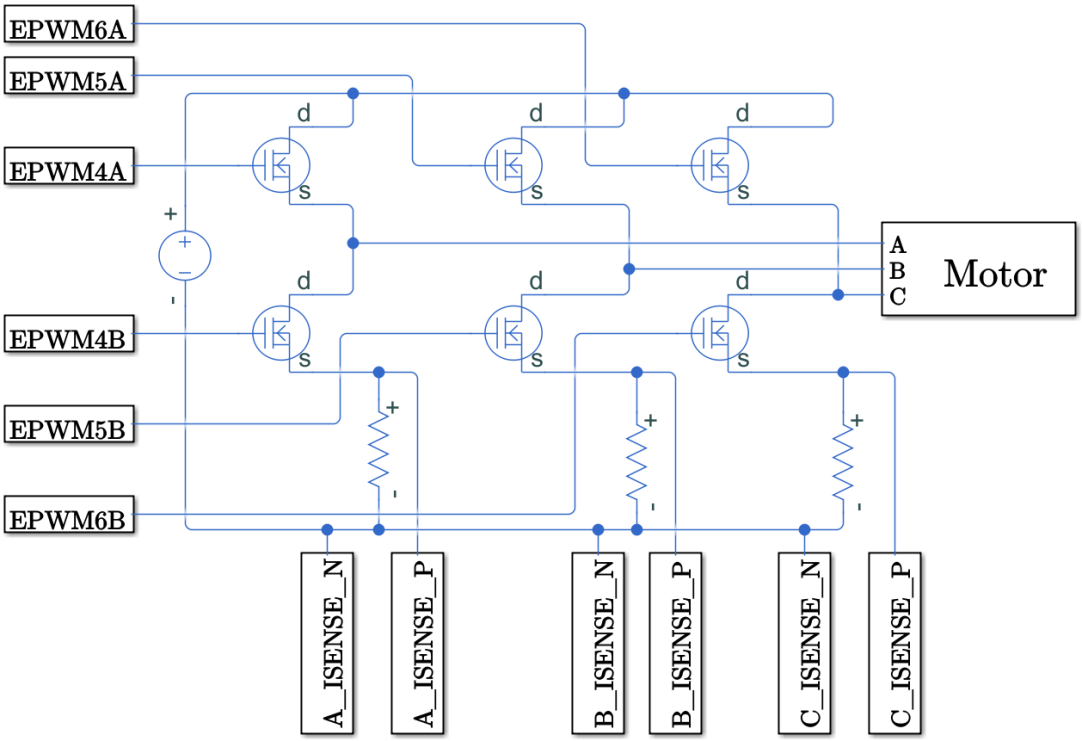
6.1 ePWM modul

O zajištění řídicích PWM signálů se stará modul *Enhanced Pulse Width Modulator* (ePWM). Každý ePWM modul má 2 kanály, EPWMxA a EPWMxB (písmeno x je využito pro obecné označení libovolného ePWM modulu, v konkrétním případě je pak nahrazeno číslem modulu, například modul ePWM4 je tvořen kanály EPWM4A a EPWM4B). Vzájemná poloha pinů na vývojové desce a na výkonovém modulu je sladěna tak, že kanál A vždy ovládá horní tranzistor příslušného půlmůstku a kanál B ovládá dolní tranzistor, viz obr 6.1. Oba kanály mají vždy společný čítač TBCTR (*Time-Base Counter*), společnou periodu TBPRD (*Time-Base Period*) a společné fázové posunutí TBPHS (*Time-Base Phase*).

Označení v dialogovém okně simulinku při nastavování parametrů bloku se v některých případech mírně liší od označení příslušných registrů v dokumentaci výrobce. Například je v některých případech vynechána část označení odpovídající názvu submodulu, kterému registr náleží (např. *Time-Base*), a registry TBCTR a TBPRD jsou označovány jako CTR (*counter*) a PRD (*period*).

Čítač TBCTR umožňuje tři různé režimy počítání – *Up*, *Down* a *Up-Down*. První dva se využívají při nesymetrickém generování PWM signálu, třetí slouží ke generování symetrického PWM signálu.

Perioda čítače TBPRD je při počítacím režimu *Up-Down* rovna polovině periody PWM signálu. Vzhledem k frekvenci procesoru 90 MHz, respektive jeho periodě $T_{\mathrm{SYSCLKOUT}} = 11,\overline{1}$ ns a zvolené frekvenci PWM signálu 20 kHZ, respektive periodě



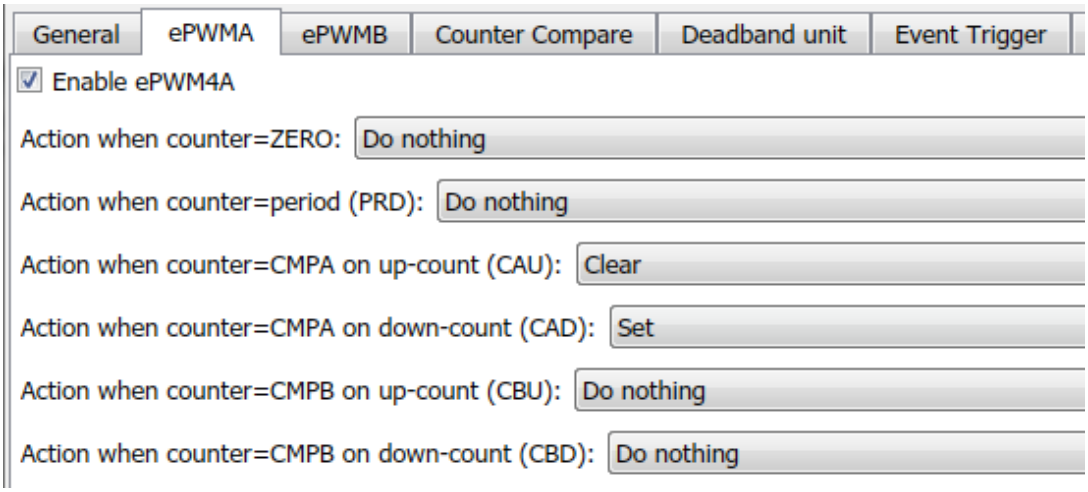
Obrázek 6.1: Zapojení motoru

$T_{PWM} = 50$ μs je možné vypočítat periodu TBPRD ze vztahu

$$\text{TBPRD} = \frac{T_{PWM}}{2 \cdot T_{\text{TBCLK}}} = \frac{50 \cdot 10^{-6}}{2 \cdot \frac{1}{90 \cdot 10^{6}}} = 2250 \tag{6.1}$$

kde T_{TBCLK} je časový úsek, který vznikne po aplikaci prescaleru na periodu cyklu procesoru $T_{\text{SYSCLKOUT}}$. Prescaler umožňuje zpomalit čítač tím, že za dobu n cyklů procesoru, kde n je celočíselná hodnota prescaleru, vytvoří jen jeden cyklus. To je vhodné v případě malých frekvencí, kde by jinak došlo k přetečení čítače TBCTR. Zároveň se tím však n–krát sníží rozlišení PWM signálu. Vzhledem k 16bitové velikosti registru TBCTR a počtu cyklů potřebných pro dosažení frekvence PWM signálu není třeba čítač zpomalovat, a proto je T_{TBCLK} v rovnici 6.1 rovno $T_{\text{SYSCLKOUT}}$.

Simulinkový blok umožňuje v závislosti na použitém režimu počítání nastavit (samostatně pro každý aktivní kanál) chování signálu ve významných časových okamžicích, viz obr 6.2. V každém z těchto okamžiků je možné nastavit, zda má dojít k nastavení logické nuly (*Clear*), logické jedničky (*Set*) nebo ponechání aktuálního stavu. Nastavení uvedené na obrázku 6.2 (pro režim *Up-Down*) je použito na kanálech EPWMxA pro všechny tři fáze motoru. Zajišťuje generování symetrického signálu, který má v oblasti kolem nulové hodnoty čítače TBCTR logickou jedničku a v oblasti kolem hodnoty čítače rovné periodě TBPRD logickou nulu. Hodnota registru CMPA (*Counter-Compare A*) vypočítaná blokem *C2000 Space Vector Generator* v tomto případě odpovídá počtu cyklů procesoru v rámci jedné periody TBPRD, kdy je signál nastaven na logickou jedničku. Takto generovaný signál je znázorněn na obrázku 3.4 vpravo (pozor na



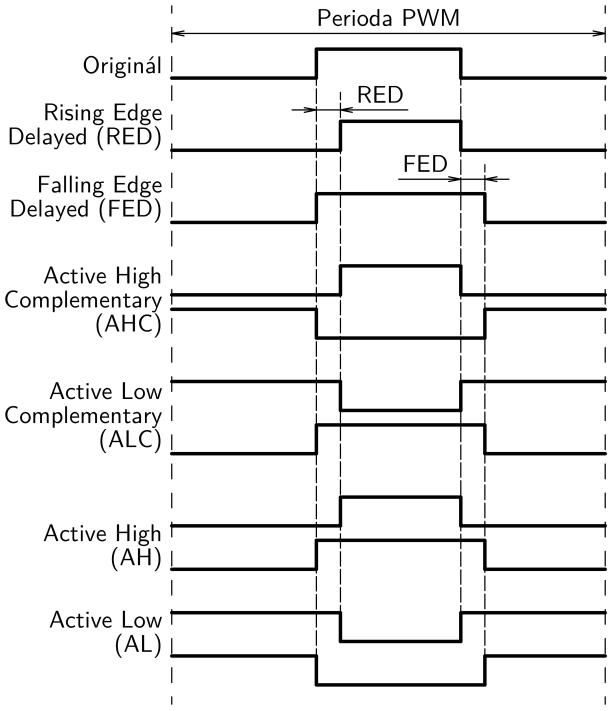
Obrázek 6.2: Použité nastavení EPWMxA

značení, na obrázku značí písmena A, B a C v závorkách u hodnot CMP danou fázi motoru, neboť v první části této práce nebyla zmínka o kanálech ePWM modulů).

Při řízení je nutné zajistit, aby nedošlo k otevření obou tranzistorů stejného půlmůstku současně, což by mělo za následek zkrat. Změna napětí řídicích PWM signálů však nikdy neproběhne okamžitě. Zároveň i otevření nebo uzavření tranzistoru nějakou dobu trvá. Proto je důležité nejdříve uzavřít otevřený tranzistor a až po určité prodlevě otevřít druhý tranzistor daného půlmůstku. Tato prodleva se nazývá mrtvá zóna (*dead-band*). V závislosti na aplikaci umožňuje ePWM modul nastavit jednu ze čtyř možností nastavení mrtvé zóny – *Active high* (AH), *Active low* (AL), *Active high complementary* (AHC) a *Active low complementary* (ALC), viz obr 6.3. Výše zmíněný požadavek potřebný v této aplikaci odpovídá nastavení AHC. Velikost mrtvé zóny je možné nastavit pro náběžnou a spádovou hranu kanálu samostatně. Jelikož nastavení mrtvé zóny jednoznačně definuje tvar PWM signálu kanálu EPWMxB na základě tvaru PWM signálu kanálu EPWMxA nebo naopak, je nutné pro správnou funkci povolit v Simulinkovém bloku pouze jeden z těchto kanálů, k aktivaci a nastavení druhého dojde automaticky. Jako zdroj signálu pro náběžnou i spádovou hranu byl v této práci vždy zvolen kanál EPWMxA a velikost obou mrtvých zón byla nastavena na 15 cyklů procesoru.

Jak již bylo zmíněno, k vývojové desce je možné připojit až dva výkonové moduly. Existují tedy dvě sady pinů, na které je možné výkonový modul nasunout. V rámci návrhu řídicího algoritmu byla zvolena sada, které odpovídají ePWM moduly 4, 5 a 6.

Pro dosažení správného efektu SVM je nutné, aby jednotlivé ePWM moduly běžely synchronizovaně. Každý modul má synchronizační vstup EPWMxSYNCI a synchronizační výstup EPWMxSYNCO. Modul ePWM1 může přijímat synchronizační vstup z externího pinu. Každý další modul má napojen synchronizační vstup na synchronizační výstup předchozího ePWM modulu (modul ePWM2 přijímá synchronizační signál od modulu ePWM1, ePWM3 od ePWM2 atd.). Každý modul může generovat synchronizační pulz na výstupu, pokud přijme synchronizační pulz na vstupu nebo se jeho čítač TBCTR rovná nule nebo CMPB (*Counter-Compare B*). Každý modul může být nastaven tak, aby při synchronizaci nastavil svůj čítač TBCTR na požadovanou



Obrázek 6.3: Možná nastavení mrtvé zóny

hodnotu fázového posunutí TBPHS, nebo synchronizační vstup ignoroval. Dále je možné nastavit směr počítání po synchronizaci (nahoru nebo dolů). V rámci této práce je modul ePWM4 nastaven tak, aby generoval synchronizační pulz při TBCTR = 0. Sám přitom synchronizační pulz nepřijímá. Moduly ePWM5 a ePWM6 jsou nastaveny tak, aby při synchronizaci nastavily čítač TBCTR na nulu. Zároveň musí modul ePWM5 generovat synchronizační pulz ve chvíli, kdy sám synchronizační pulz přijme.

K dispozici je ještě řada dalších funkcí ePWM modulů. V této práci je však využita už jen možnost spustit konverzi ADC převodníku. Každý ePWM modul má možnost vyvolat konverzi ADC převodníku při určitých událostech. Dále je možné nastavit, aby k vyvolání konverze došlo pokaždé, nebo vždy až při druhé, případně třetí události. Vzhledem k nastavení ePWM modulů a principu měření proudů popsanému v kapitole 6.2 je konverze vyvolávána modulem ePWM4 pokaždé, když se čítač TBCTR rovná periodě TBPRD. [10] [38]

6.2 Měření proudů ADC převodníkem

6.2.1 Princip měření proudů

Při řízení PMSM motorů vektorovou metodou je zapotřebí měřit proudy protékající alespoň dvěma ze tří fází motoru. Toho lze dosáhnout pomocí měřicích odporů. Měřicí odpory jsou velmi malé a velmi přesné odpory umístěné v sérii mezi dolními tranzistory a nulou zdroje, jak je znázorněno na obrázku 6.1. V případě, že je dolní tranzistor otevřený, protéká skrz příslušný odpor stejný proud jako skrz vinutí dané fáze motoru.

Při známé velikosti elektrického odporu lze při změření napětí na měřicím odporu vypočítat protékající proud.

Měření je třeba provádět pokud možno současně. Pokud to možné není a měření proudů probíhá postupně, je třeba proud měřit ihned po dokončení předchozího měření (na jiné fázi). S ohledem na tuto skutečnost a s ohledem na fakt, že proud měřicím odporem protéká pouze, pokud je otevřen dolní tranzistor, je zřejmé, že při měření musí být všechny měřené fáze připojeny na nulu. Vzhledem k nastavení ePWM modulů uvedenému v kapitole 6.1 je tohoto stavu dosaženo v okolí hodnoty časovače TBCTR rovné periodě čítače TBPRD. Proto je konverze ADC převodníku vyvolávána právě při této hodnotě časovače.

Měření proudů je při tomto zapojení možné provádět pomocí dvou nebo pomocí všech tří měřicích odporů. První varianta je méně náročná na výpočetní výkon, neboť neobsahuje rozhodovací operace. Má však jedno omezení. Pokud se velikost korekčního napětí $\vec{v}^{ref}$ blíží maximu nebo je maximální, není při některých úhlech vektoru napětí možné měřit všechny tři proudy, neboť jedna z fází je připojena na nulu příliš krátkou dobu nebo není v extrémním případě na nulu připojena vůbec. Pokud by se zrovna jednalo o fázi, na které měření neprobíhá, nebyl by to problém. Prakticky však tento případ nastává pouze při určitých úhlech. Proto je třeba při použití metody se dvěma měřicími odpory omezit velikost korekčního napětí tak, aby byla vždy zajištěna dostatečně dlouhá doba na provedení měření. V případě měření proudů pomocí všech tří měřicích odporů není nutné velikost vektoru napětí takto omezovat. Po provedení měření lze na základě známého úhlu natočení rotoru zvolit, které dva změřené proudy využít k výpočtu. [39]

6.2.2 Konstrukce výkonového modulu

Napětí vznikající na měřicím odporu je před měřením analogově zesíleno. Integrovaný obvod DRV8301 nacházející se na výkonovém modulu umožňuje zpracování napěťových signálů pouze ze dvou měřicích odporů. Na výkonovém modulu je zapojen tak, že zpracovává napětí na měřicích odporech odpovídajících fázím A a B. Pro zpracování napěťových signálů na měřicím odporu fáze C je proto k dispozici dodatečný obvod integrovaný přímo na výkonovém modulu. Hlavním rozdílem zde přitom je, že obvody uvnitř DRV8301 jsou složitější a umožňují zesílení napětí 10krát, 20krát, 40krát nebo 80krát. Obvod pro fázi C zesiluje napětí vždy 10krát. Zesílené napětí je následně odečítáno od poloviny referenčního napětí V_{REF}, čímž vznikne napětí V_O. Referenční napětí je do DRV8301 přivedeno pomocí vstupního pinu REF. Vzhledem k tomu, že proud může fází protékat oběma směry, je na tento pin přivedeno maximální napětí, které dokáže vyhodnotit ADC převodník vývojové desky, tedy 3,3 V. Stejné napětí je přivedeno i do obvodu fáze C. Napětí přivedené na vstup ADC převodníku lze vzhledem k zapojení obvodů vypočítat ze vztahu

$$V_O = \frac{V_{REF}}{2} - G \cdot (\text{x_ISENSE_P} - \text{x_ISENSE_N}) \tag{6.2}$$

kde G je zesílení napětí a x_ISENSE_P a x_ISENSE_N jsou napětí před a za měřicím odporem, viz obr. 6.1. Písmeno x v tomto případě zastupuje danou fázi A, B nebo C.

Z uvedeného vyplývá, že napětí $V_O = 1,65$ V odpovídá nulovému proudu, napětí $V_O = 3,3$ V odpovídá maximálnímu měřitelnému proudu tekoucímu v kladném směru

(za kladný je považován proud tekoucí vinutím fáze od zdroje do motoru) a $V_O = 0$ V odpovídá maximálnímu měřitelnému proudu tekoucímu v záporném směru. Maximální měřitelná velikost proudu je v obou směrech stejná a závisí na velikosti elektrického odporu měřicích odporů a na zesílení napětí G. Elektrický odpor měřicích odporů je na všech fázích v tomto případě $R_B = 0,01\ \Omega$. Velikost proudu i_x protékajícího danou fází x proto lze určit ze vztahu

$$i_\mathrm{x} = \frac{V_O - \frac{V_{REF}}{2}}{G \cdot R_B} = \frac{V_O - 1,65}{0,01 \cdot G} \tag{6.3}$$

Maximální měřitelné proudy při jednotlivých zesíleních jsou uvedeny v tabulce 6.1.

G	$i_{\mathrm{x}_{max}}$	Rozlišení
10	16,5 A	8 mA
20	8,25 A	4 mA
40	4,125 A	2 mA
80	2,0625 A	1 mA

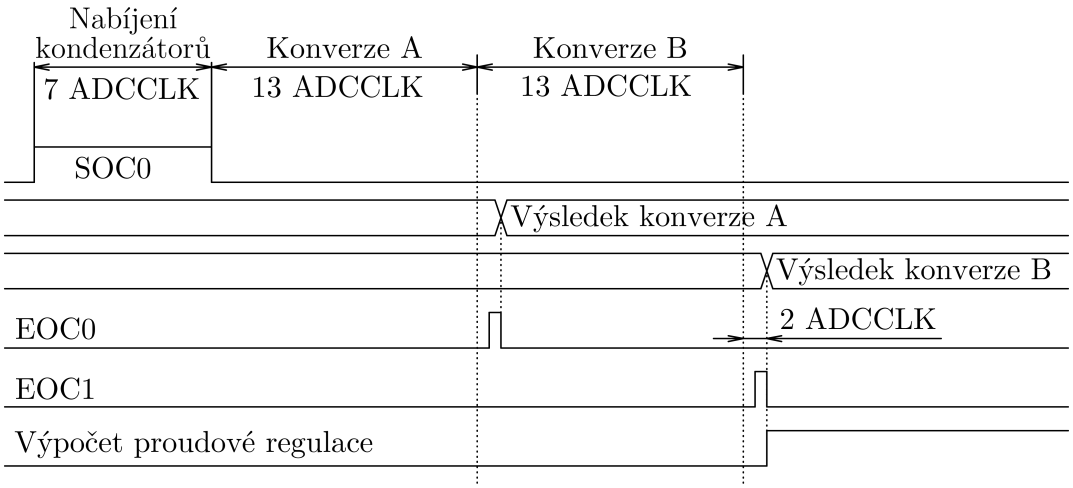
Tabulka 6.1: Maximální měřitelné proudy a proudová rozlišení v závislosti na použitém zesílení napětí

Použitý motor je dimenzován na jmenovitý proud $I_n = 2,1\ A_{rms}$. Tomu odpovídá proudová špička 3 A. Maximálně lze tedy použít zesílení $G = 40$. V závislosti na rozlišení ADC převodníku se se zesílením mění i rozlišení měřeného proudu. Použité ADC převodníky jsou 12bitové (rozlišení 4096 na celý rozsah napětí 0 V – 3,3 V). Z toho vyplývají proudová rozlišení pro jednotlivá zesílení, viz tab. 6.1.

Jak již bylo zmíněno, zesílení lze zvolit pouze pro fáze A a B. Nicméně vzhledem k tomu, že je v této práci motor řízen v nezatíženém stavu, není třeba využívat maximální možnou velikost vektoru korekčního napětí $\vec{v}^{\,ref}$. Díky tomu není žádný důvod měřit proudy pomocí všech tří měřicích odporů. Pro měření jsou proto vždy použity měřicí odpory fází A a B a pomocí SPI komunikace mezi vývojovou deskou a výkonovým modulem je při spuštění řídicího algoritmu nastavena hodnota zesílení $G = 40$, viz kapitola 7.2.1. [10] [26] [28] [40]

6.2.3 ADC modul

Samotné měření zesíleného napětí zajišťuje ADC (*Analog-to-Digital Converter*) modul. Vývojová deska obsahuje jeden 12bitový ADC modul se dvěma S+H (*sample-and-hold*) obvody. Na vstup každého z těchto dvou obvodů může být přivedeno napětí ze sedmi různých pinů vývojové desky. Jednotlivým obvodům přísluší odlišné piny, celkem lze tedy k měření napětí ADC převodníkem použít až čtrnáct pinů vývojové desky. Konverzi napětí nelze provádět současně z více pinů, přesto však lze díky dvěma S+H obvodům zajistit současné měření dvou proudů. Napětí však v takovém případě musí být přivedeno na piny ADCINxA a ADCINxB se stejným číslem x. Napětí z pinů označených ADCINxA je vždy přivedeno na první S+H obvod, napětí z ADCINxB pinů je přiváděno do druhého S+H obvodu. Vzájemná poloha pinů vývojové desky



Obrázek 6.4: Časové schéma měření proudů

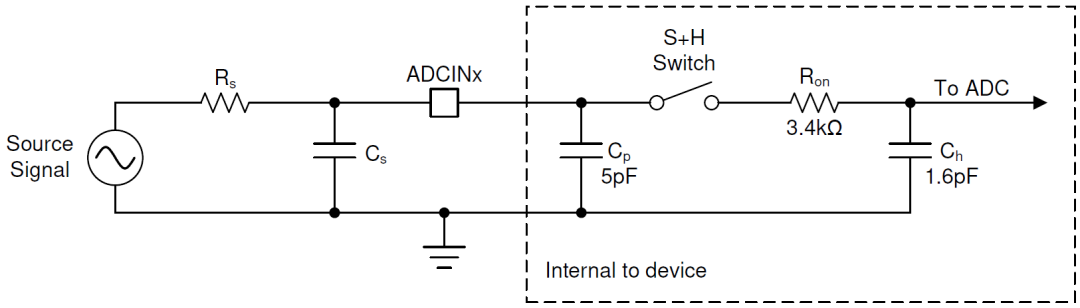
a výkonového modulu je sladěna tak, že je možné současně měřit proudy fází A a B pomocí pinů ADCIN3A a ADCIN3B, čehož je v této práci využito.

Měření začíná vyvoláním SOC (*Start-Of-Conversion*) pulzu, viz obr 6.4. Ten může být vyvolán programem nebo hardwarově (například ePWM modulem). Při náběžné hraně tohoto pulzu je sepnut S+H spínač a tím je přivedeno napětí na kondenzátor C_h, viz obr 6.5. Ten se po dobu trvání SOC pulzu nabíjí. Se spádovou hranou SOC pulzu dojde k odpojení vstupního napětí rozpojením S+H spínače a je zahájena konverze napětí udržovaného kondenzátorem C_h. Pokud je nastavený simultánní režim, je po skončení konverze na kondenzátoru v prvním S+H obvodu zahájena konverze napětí na kondenzátoru v druhém S+H obvodu.

V souvislosti s každou z konverzí je dále generován EOC (*End-Of-Conversion*) pulz. V simultánním režimu jsou proto vyvolány vždy dva, první (se sudým číslem) je vyvolán v souvislosti s konverzí kanálu A, která vždy probíhá jako první, a druhý (s následujícím lichým číslem) je vyvolán v souvislosti s konverzí kanálu B. V závislosti na nastavení je EOC pulz generován buď na začátku konverze daného kanálu nebo na konci. Kterýkoli z těchto dvou pulzů může být nastaven na vyvolání hardwarového přerušení. V Simulinku se možnost tohoto nastavení neobjevuje, analýzou vygenerovaného C kódu však lze zjistit, že je využíván až druhý pulz. V této práci je nastaveno generování EOC pulzů na konci konverze a pulz EOC1 vyvolává hardwarové přerušení, které spouští výpočet proudové regulační smyčky, viz obr. 6.4.

SOC pulz musí být dostatečně dlouhý, aby došlo k dostatečnému nabití kondenzátoru C_h. Tato doba závisí na parametrech prvků S+H obvodu a na parametrech případného odporu R_s a případného kondenzátoru C_s nacházejících se mezi zdrojem měřeného napětí a vstupem ADCINx do S+H obvodu. Minimální nastavitelná velikost SOC pulzu je však 7 cyklů hodin (*clock cycles*) ADC modulu. Samotná konverze pak probíhá vždy 13 cyklů hodin (ADCCLK) ADC modulu.

Za zesilovačem měřeného napětí se v případě obvodu DRV8301 nachází odpor 100 Ω,



Obrázek 6.5: Zapojení zdroje měřeného napětí [10]

kondenzátor však nikoli. Na základě výpočtů

$$\begin{aligned}\tau &= (R_s + R_{on}) \cdot C_h + R_s \cdot (C_s + C_p) = \\ &= (100 + 3400) \cdot 1,6 \cdot 10^{-12} + 100 \cdot \left(0 + 5 \cdot 10^{-12}\right) = \\ &= 6,1 \cdot 10^{-9}\ \text{s}\end{aligned} \tag{6.4}$$

$$\begin{aligned}k &= \ln\left(\frac{2^n}{chyba\ ustálení}\right) - \ln\left(\frac{C_s + C_p}{C_h}\right) = \\ &= \ln\left(\frac{2^{12}}{\frac{1}{2}}\right) - \ln\left(\frac{0 + 5 \cdot 10^{-12}}{1,6 \cdot 10^{-12}}\right) = \\ &= 7,871\end{aligned} \tag{6.5}$$

$$t_{s+h} = k \cdot \tau = 7,871 \cdot 6,1 \cdot 10^{-9} = 4,801 \cdot 10^{-8}\ \text{s} \tag{6.6}$$

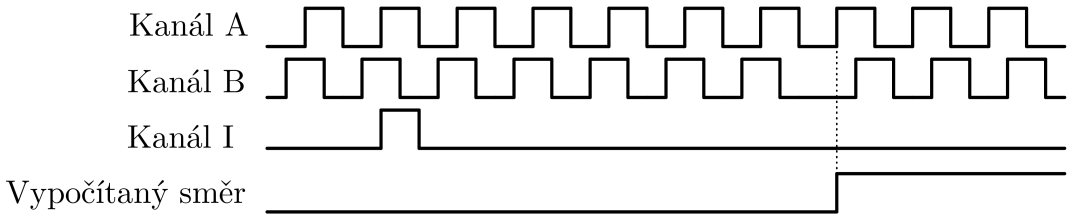
$$\text{ACQPS} = (t_{s+h} \cdot f) - 1 = \left(4,801 \cdot 10^{-8} \cdot 45 \cdot 10^{6}\right) - 1 = 1,16 \tag{6.7}$$

provedených dle dokumentace [10] je potřebná délka SOC okolo dvou cyklů hodin ADC modulu. Proto byla ponechána nejmenší možná hodnota délky SOC pulzu 7 cyklů hodin ADC modulu. Ve výpočtech je f frekvence hodin ADC modulu dosažená uplatněním prescaleru o velikosti 2 na hodiny procesoru. Registr ACQPS udává délku SOC pulzu a jeho hodnota (celé číslo) je vždy o jedna menší než velikost SOC pulzu v cyklech hodin ADC modulu. [10] [28] [41]

6.3 eQEP modul

Použitý motor má integrovaný rotační inkrementální enkodér. Jeho vyhodnocení zajišťuje eQEP (*enhanced Quadrature Encoder Pulse*) modul. Každý eQEP modul má vlastních pět pinů na vývojové desce (nula, napájení, kanál A, kanál B a kanál I).

Kanál I je určen pro indexační pulz, který umožňuje určení absolutní pozice inkrementálního enkodéru. V případě rotačních enkodérů je generován jednou za otáčku. Význam kanálů A a B závisí na nastavení eQEP modulu. Některé inkrementální enkodéry generují dva obdélníkové signály vzájemně posunuté o 90° reprezentující



Obrázek 6.6: Průběh signálů enkodéru pro variantu *Quadrature-clock Mode* a určení směru

pulzy enkodéru ve dvou samostatných řadách, viz obr. 6.6. Vzájemné posunutí slouží k určení směru pohybu. Jiné enkodéry jedním kanálem poskytují informaci o směru a druhým obdélníkový signál reprezentující pulzy enkodéru. V obou těchto případech lze počítat relativní polohu i při obousměrném provozu. První varianta je v eQEP modulu označována jako *Quadrature-clock Mode* a druhá jako *Direction-count Mode*. Při druhé variantě vyžaduje eQEP modul obdélníkový signál na pin EQEPA a informaci o směru na pin EQEPB. Je možné také nastavit režimy *UP-count mode* a *DOWN-count mode*. Ty umožňují přičítání pulzů pouze v jednom směru.

Použitý enkodér vyžaduje nastavení *Quadrature-clock Mode*. Jeho rozlišení je 2048 pulzů na otáčku na každý kanál, což při počítání náběžných i spádových hran vede k rozlišení 8192 poloh na otáčku. Naneštěstí má poškozený indexační pulz a ten proto není možné využít. Místo toho byl k určení absolutní polohy vytvořen postup popsaný v kapitole 7.2, který je proveden při každém spuštění řídicího algoritmu. Tento postup umožní resetovat v definované poloze polohový čítač QPOSCNT na inicializační hodnotu QPOSINIT. Registr polohového čítače QPOSCNT je 32bitová celočíselná proměnná (v Simulinku označená *uint32*, viz kapitola 7.1.1) s rozsahem 0 až $(2^{32} - 1)$. Maximální hodnotu čítače však lze omezit na hodnotu QPOSMAX. Tím je možné regulovat, při jaké hodnotě čítač přeteče. To se může hodit, pokud je například zapotřebí, aby k přetečení došlo na hodnotě, která není mocninou dvou, nebo pokud je proměnnou nutné následně převést na jiný datový typ s menším rozsahem kladných čísel. Aby se zjednodušily některé výpočty, je před nimi v této práci proměnná převedena na typ *int32*, který má poloviční rozsah v kladných číslech. Polohová regulace sice nepočítá s možností přetečení polohového čítače a celkově ani s tak velkými hodnotami, i tak je ale omezení nastaveno na hodnotu $2^{31} - 1$. Aby byl zachován co největší rozsah do obou směrů, byla inicializační hodnota QPOSINIT zvolena uprostřed použitého rozsahu, tedy 2^{30}. Důležité zde také je, že tato hodnota je bezezbytku dělitelná rozlišením enkodéru 2^{13}. To usnadní potřebné výpočty.

Všechny funkce eQEP modulu probíhají na hardwarové úrovni a nezasahují do aktuálně běžícího programu. To umožňuje správnou a dostatečně rychlou detekci pulzů. Hodnoty z registrů, jako je registr čítače polohy QPOSCNT, je možné kdykoli programově přečíst a získat tak aktuální polohu. Modul má ještě řadu dalších funkcí, ty však nebyly v této práci využity. [10] [42]

Kapitola 7

Softwarové řešení řídicího algoritmu

7.1 Základní informace

7.1.1 Datové typy

Důležitou roli při návrhu řídicího algoritmu hrají datové typy proměnných. Simulink, ve kterém je návrh prováděn, poskytuje velké množství různých datových typů. Výchozím datovým typem, který Simulink používá, je 64bitový formát s plovoucí desetinnou čárkou nazývaný *double*. Další formáty s plovoucí desetinnou čárkou jsou *single* (32bitový) a *half* (16bitový). Simulink dále umožňuje použití celočíselných proměnných *int8*, *uint8*, *int16*, *uint16*, *int32*, *uint32*, *int64* a *uint64*. Čísla v názvech datových typů označují velikost datových typů v bitech. Zkratka *int* označuje *signed* (včetně znaménka) *integer* a *uint* označuje *unsigned* (bez znaménka, pouze nezáporná čísla) *integer*. Dále je možné využívat logické proměnné typu *boolean* a textové řetězce v datovém typu *string*.

Procesor vývojové desky má vestavěnou podporu pro datové typy s plovoucí desetinnou čárkou. Podporuje však pouze 16bitové a 32bitové proměnné. Pokud je v Simulinkovém modelu použita proměnná typu *int8* nebo *uint8*, nahradí ji toolbox *Embedded Coder®* generující kód odpovídající 16bitovou proměnnou. Při simulaci je však využita nastavená 8bitová proměnná. To vede k rozdílnému chování simulovaného modelu oproti výslednému programu běžícímu na vývojové desce v případě, že dojde k přetečení. Proto je důležité se při návrhu 8bitovým formátům vyvarovat. To samé však platí pro výchozí 64bitový datový typ *double*. Ten je totiž při generování kódu převeden na 32bitový typ *single*.

Některé bloky určené pro generování optimalizovaného kódu vyžadují použití datových typů s pevnou desetinnou čárkou. Práci s takovýmito proměnnými umožňuje v simulinku toolbox *Fixed-Point Designer™*. Existuje několik způsobů, jak může být hodnota pomocí těchto datových typů kódována. Aby však bylo možné z modelu generovat kód pro procesor vývojové desky, je využita možnost definice přes délku proměnné v bitech (*WordLength*) a počet bitů za desetinnou čárkou (*FractionLength*). K vytvoření takovýchto proměnných je k dispozici funkce *fixdt* se syntaxí

$$fixdt(Signed, WordLength, FractionLength) \tag{7.1}$$

kde *Signed* určuje, zda bude jeden bit proměnné vyhrazen pro znaménko či nikoli (0 – *unsigned*, 1 – *signed*). Hodnotu *RealWorldValue*, kterou proměnná takovéhoto datového

typu reprezentuje, lze vyjádřit vztahem

$$RealWorldValue = StoredInteger \cdot 2^{-FractionLength} \tag{7.2}$$

kde *StoredInteger* je skutečně uložená hodnota. Při návrhu datového typu je opět důležité, aby velikost proměnné *WordLength* byla 16 nebo 32 bitů. Funkce *fixdt* vytváří v Simulinku objekt *Simulink.NumericType* zastupující datový typ, v této práci však bude k popisu datových typů s pevnou desetinnou čárkou využívána přímo funkce *fixdt* s použitými parametry. [43] [44] [45] [46] [47] [48]

7.1.2 Optimalizace kódu

V rámci použitého balíčku hardwarové podpory pro procesory C2000 od TI jsou k dispozici také knihovny pro tvorbu optimalizovaného kódu. Konkrétně se jedná o knihovnu matematických operací *C28x IQmath* a knihovnu pro digitální řízení motorů *C28x DMC*. Bloky z těchto knihoven však pracují jen s datovými typy s pevnou desetinnou čárkou. Navíc musí být použitý datový typ 32bitový včetně znaménka a počet bitů za desetinnou čárkou *FractionLength* musí být v požadovaném rozsahu, který je pro různé bloky různý. Pro bloky z DMC knihovny využité v této práci je tento rozsah nastaven od 1 do 29 včetně. [49]

7.1.3 Úlohy běžící na různých frekvencích

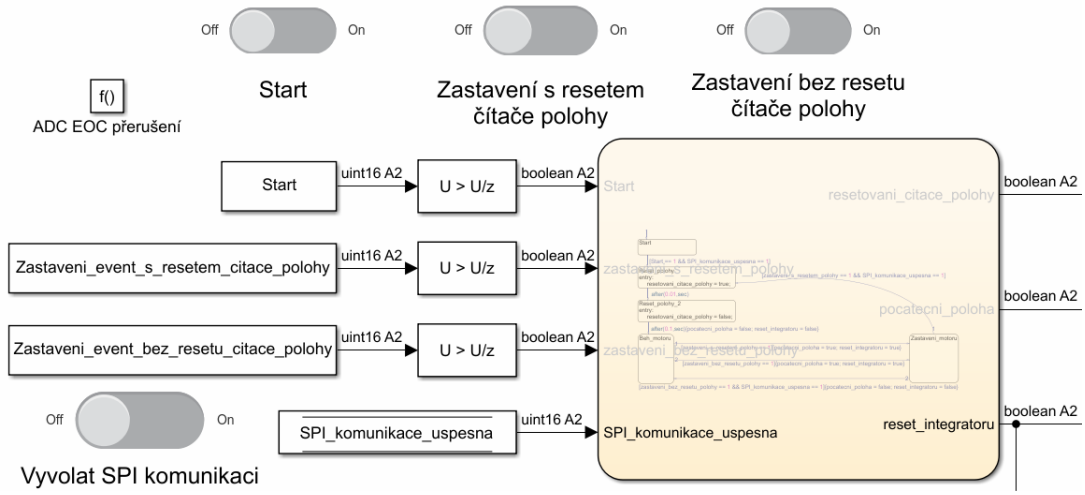
Simulink umožňuje počítat různé části modelu s různými časovými kroky numerického řešiče. V takovém případě rozdělí model na různé úlohy podle použitého časového kroku. V rámci modelu však ve spoustě případů není možné přímo spojit bloky počítané s různými časovými kroky, protože by Simulink nevěděl, jakým způsobem má signál převést. Proto je k převodu nutné použít blok *Rate Transition*, pomocí kterého se vlastnosti převodu nastaví.

Časový krok se v Simulinku nastavuje pomocí *sample time*. Jedná se o vektor dvou hodnot, z nichž první určuje periodu vzorkování (časový krok) a druhá počáteční časové posunutí. Pokud je však časové posunutí nulové, není třeba ho zadávat. Většina bloků, které umožňují nastavení časového kroku, má možnost *sample time* zdědit od ostatních bloků.

Existuje několik typů časových kroků. Kromě použití pravidelných diskrétních kroků je možné například výpočet části modelu spouštět programově pomocí funkce, například při splnění určitých podmínek. Toho také využívá blok hardwarového přerušení. Pro přehlednost je také automaticky změněna barva bloků a signálů podle toho, ke které časové úloze patří. Legenda barev a označení jednotlivých úloh pro použitý model je zobrazena na obrázku 7.1. Barevné odlišení lze vypnout, čehož je v tomto případě využito pro tvorbu schémat do příloh, neboť v tištěné verzi nejsou texty v některých barvách čitelné. Označení úlohy je u signálů ve schématech uvedeno vždy za datovým typem. [50] [51]

Discrete		Period
	D1	1.0000e-003
	D2	0.1
	D3	0.5
Event		
	A1	Asynchronous 1
	A2	Asynchronous 2
Other		
	Inf	Constant
	M	Multirate

Obrázek 7.1: Použité časové kroky

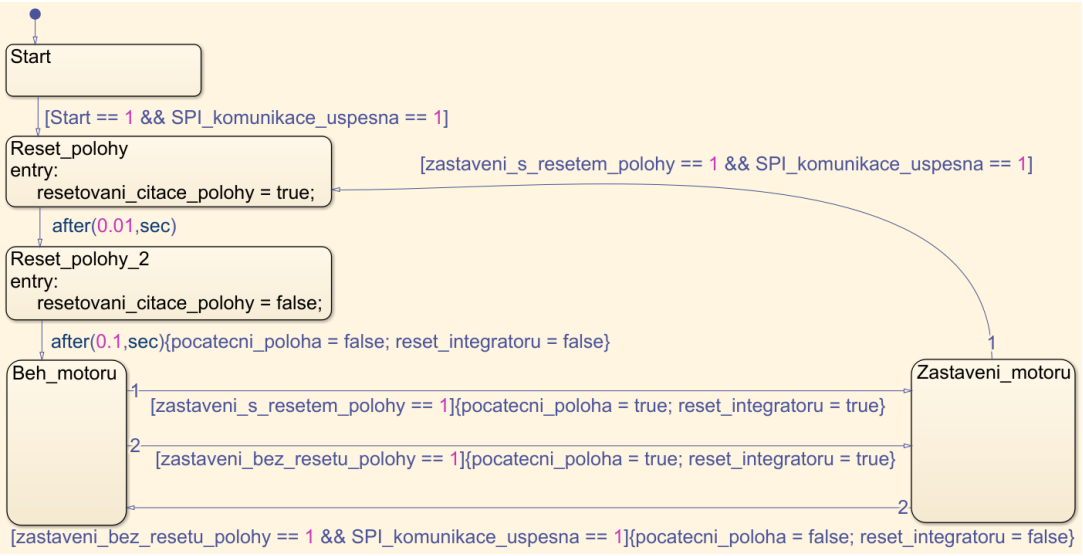


Obrázek 7.2: Ovládání motoru

7.2 Inicializační rutina a ovládání motoru

Jak již bylo zmíněno, enkodér použitého motoru má poškozený kanál s indexačním pulzem. Proto je při spuštění řídicího algoritmu vždy nejdříve generován vektor napětí ve směru osy fáze A. Ten zajistí natočení motoru do polohy, kde se úhel theta rovná nule. Následně je možné změnou jednoho parametru z logické 0 na logickou 1 spustit samotné řízení. Model je navržen tak, že je spuštění řízení motoru provedeno přepínačem přímo v Simulinku, který komunikuje s vývojovou deskou pomocí sériové komunikace přes XCP protokol, viz obrázek 7.2. Bylo by však snadno možné vyměnit blok s parametrem, na který je vázaný přepínač, za digitální vstup a spustit tak řízení i bez připojení k počítači. Muselo by se pouze připojit na zvolený pin nějaké tlačítko.

Dále byly do modelu zavedeny další dva přepínače. Umožňují motor zastavit přepnutím hodnot vstupujících do registrů ePWM modulů zpět na konstanty tvořící vektor napětí ve směru osy fáze A. Následně je možné řídicí algoritmus opět spustit, a to



Obrázek 7.3: Schéma Stateflow

buď s opětovným resetováním čítače polohy, nebo bez něj. Přepínače zde plní funkci tlačítek. Model reaguje vždy pouze na změnu z logické nuly na jedničku, nikoli naopak.

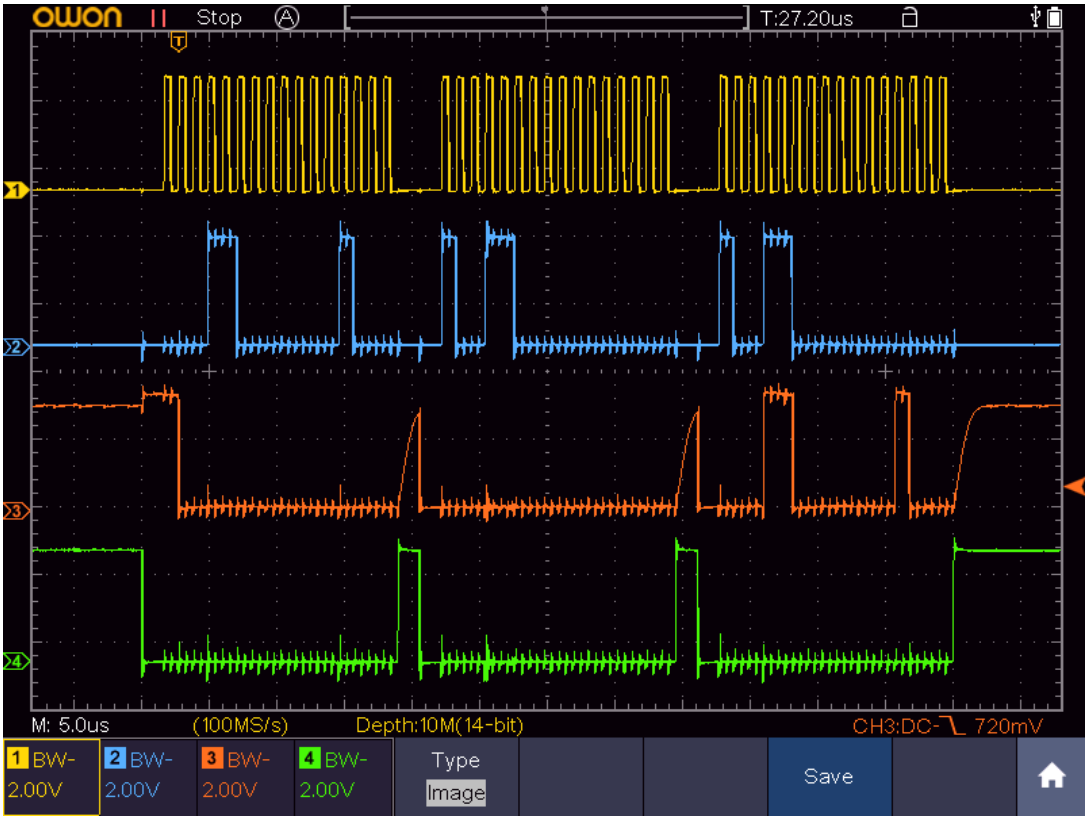
Vzhledem k tomu, že jsou výpočty regulačních smyček prováděny i při stojícím motoru, je třeba před roztočením motoru vždy resetovat integrátory regulátorů na nulu. Pro jednoduchost jsou integrátory resetovány opakovaně po celou dobu, kdy je motor zastaven. Dále je nutné při resetování čítače polohy nastavit nulovou hodnotu rychlosti. Čítač polohy eQEP modulu je totiž při spuštění programu nastaven na hodnotu 0. Při jeho resetování na hodnotu 2^{30} by tak výpočtem vyšla nesmyslná rychlost. Ta sice v okamžiku resetu čítače není zohledněna, nicméně filtr rychlosti by takto velkou změnu promítl do následného řízení a došlo by k velkému cuknutí motoru.

Ovládací logika je vytvořena pomocí toolboxu *Stateflow*. Ten umožňuje vytvořit různé stavy a při splnění určitých podmínek mezi nimi přepínat. Schéma použitých stavů a podmínek je zobrazeno na obrázku 7.3.

7.2.1 SPI komunikace

Před roztočením motoru je dále nutné zajistit správné nastavení výkonového modulu. Zesílení napětí na měřicích odporech je totiž ve výchozím stavu nastaveno na hodnotu 10. Nastavení výkonového modulu se provádí pomocí SPI komunikace mezi vývojovou deskou (*master*) a výkonovým modulem (*slave*). Komunikace probíhá pomocí čtyř pinů, viz obr. 7.4. Pin SCS (zelená) odděluje cykly komunikace. V rámci jednoho cyklu komunikace přijme výkonový modul 16 bitů (modrá) a zároveň 16 bitů odešle (červená). Aby mohl výkonový modul správně vyhodnotit napětí na vstupním pinu SDI a správně nastavit napětí na výstupním pinu SDO, poskytuje mu vývojová deska hodiny střídáním napětí na pinu SCLK (žlutá).

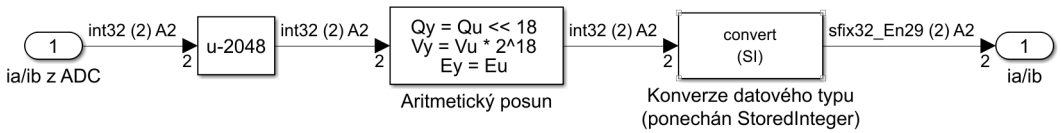
Data, která výkonový modul přijímá, musí mít správnou strukturu. První bit určuje, zda jde o požadavek o zápis či o čtení. Následují 4 bity s adresou registru a 11 bitů s daty. Pokud jde o zápis, vrátí výkonový modul v následujícím cyklu obsah prvního



Obrázek 7.4: Nastavení výkonového modulu pomocí SPI komunikace

registru s diagnostickými daty. Pokud jde o čtení, vrátí v následujícím cyklu obsah požadovaného registru. Pro přepsání jednoho registru a jeho následnou kontrolu je proto zapotřebí tří cyklů komunikace.

Nastavení zesílení napětí na měřicích odporech je uloženo ve dvou bitech posledního registru. Blok Simulinku zprostředkovávající SPI komunikaci vyžaduje jako vstup číslo ve formátu *uint16*, které je reprezentováno požadovanými bity. Změně nastavení požadovaného zesílení na hodnotu 40 se současným ponecháním výchozích hodnot ostatních bitů daného registru odpovídá číslo 6 152. Následnému přečtení této adresy odpovídá číslo 38 912. To je posláno dvakrát, aby mohla být přijata odpověď, viz obr. 7.4. Požadované odpovědi odpovídá číslo 6 152. Řídicí algoritmus je vytvořen tak, že pokud číslo přijaté v posledním cyklu neodpovídá požadované hodnotě, je celý proces vyvolán znovu. K tomu může dojít například, pokud výkonový modul ještě není připojen ke zdroji napětí, které má distribuovat do motoru (napájení přes piny z vývojové desky nestačí). V takovém případě odmítá výkonový modul komunikovat. S tím také souvisí, že jsou při odpojení od tohoto napájení resetovány registry do výchozích hodnot. Proto je přidána možnost vyvolat komunikaci také manuálně pomocí přepínače. Aby nedocházelo k rušení komunikace vyvoláním hardwarového přerušení pro výpočet proudové regulace, je subsystém zajišťující SPI komunikaci vyvoláván také hardwarovým přerušením, ovšem s větší prioritou. [27] [28] [52]



Obrázek 7.5: Škálování změřených proudů

7.3 Regulační smyčky

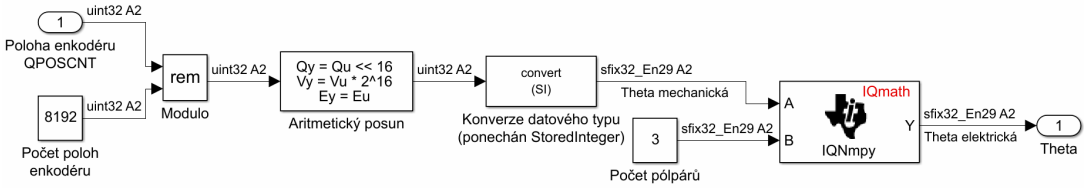
7.3.1 Proudová regulační smyčka

Výpočet hodnot pro registry ePWM modulů se skládá z několika částí. Nejdříve je nutné naškálovat změřené vstupní hodnoty (proudy A a B a poloha). Následují Clarkeové transformace a Parkova transformace. Jejich výpočet zajišťují příslušné bloky z *C28x DMC* knihovny, viz schéma v příloze. Následuje regulační výpočet pomocí dvou sériových regulátorů, jejichž model je sestaven tak, aby byly zajištěny dynamické limity integrátoru dle kapitoly 4.5. Model regulátoru je zobrazen v příloze. Vzhledem k omezením vyplývajícím z měření proudů dvěma měřicími odpory je výstup obou regulátorů omezen na hodnotu ±0,5. Dále je provedena inverzní Parkova transformace a generace hodnot pro registry ePWM modulů pomocí bloků z knihovny *C28x DMC*. Nakonec je nutné naškálovat vygenerované hodnoty na výsledné hodnoty CMPA jednotlivých EPWMxA kanálů.

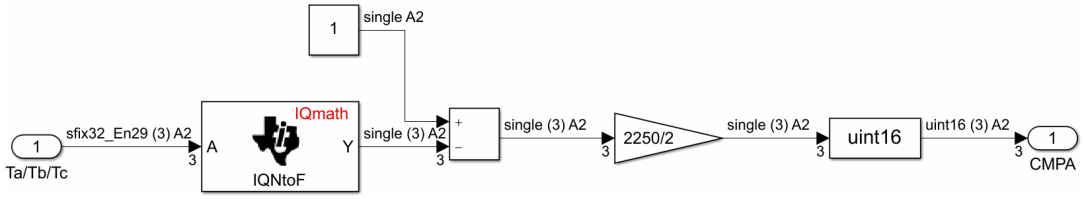
Škálování vstupů

Změřené proudy je nutné nejdříve naškálovat. Výstupem ADC převodníku jsou celočíselné hodnoty v intervalu $\langle 0; 4095 \rangle$. Jak již bylo zmíněno, proud může protékat vinutím oběma směry a nulový proud se nachází na středu měřicího rozsahu. Proto je nejdříve od změřené hodnoty odečtena polovina rozsahu (hodnota 2048). Kladné hodnoty nyní představují proudy tekoucí v kladném směru a naopak. Následně je zajištěna změna datového typu na typ s pevnou desetinnou čárkou *fixdt(1,32,29)* (ve schématech označen též *sfix32_En29*) současně s přepočítáním hodnot na rozsah $\langle -1; 1 \rangle$. Subsystém Simulinkového modelu zajišťující popsané škálování je zobrazen na obrázku 7.5 a v příloze.

Pro výpočet Parkovy transformace je dále nutné znát úhel natočení rotorového souřadného systému vůči statorovému souřadnému systému θ. Funkce *_IQsinPU* a *_IQcosPU*, které jsou využity při výpočtu Parkovy a zpětné Parkovy transformace, vyžadují úhel θ v tzv. *per-unit radians*, kde úhel v rámci jedné elektrické otáčky leží v rozsahu $\langle 0; 1 \rangle$. Toho lze dosáhnout vhodným škálováním změřené polohy v kombinaci s přičtením počátečního úhlu θ_0. Jak již bylo zmíněno, výstupem eQEP modulu je poloha enkodéru QPOSCNT v rozsahu $\langle 0; 2^{31} - 1 \rangle$ s výchozí hodnotou 2^{30}. Výstupní hodnota eQEP modulu je nejprve podrobena funkci modulo, která vrací zbytek po celočíselném dělení hodnotou 8192 (počet poloh enkodéru na mechanickou otáčku). Výsledek, který se nyní nachází v rozsahu $\langle -8191; 8191 \rangle$, je pak převeden do rozsahu $(-1; 1)$ opět v datovém typu *fixdt(1,32,29)*. Vzhledem k tomu, že použitý motor je šestipólový, odpovídá jedna mechanická otáčka třem elektrickým otáčkám. Následuje



Obrázek 7.6: Škálování úhlu θ



Obrázek 7.7: Škálování výstupu proudové regulace

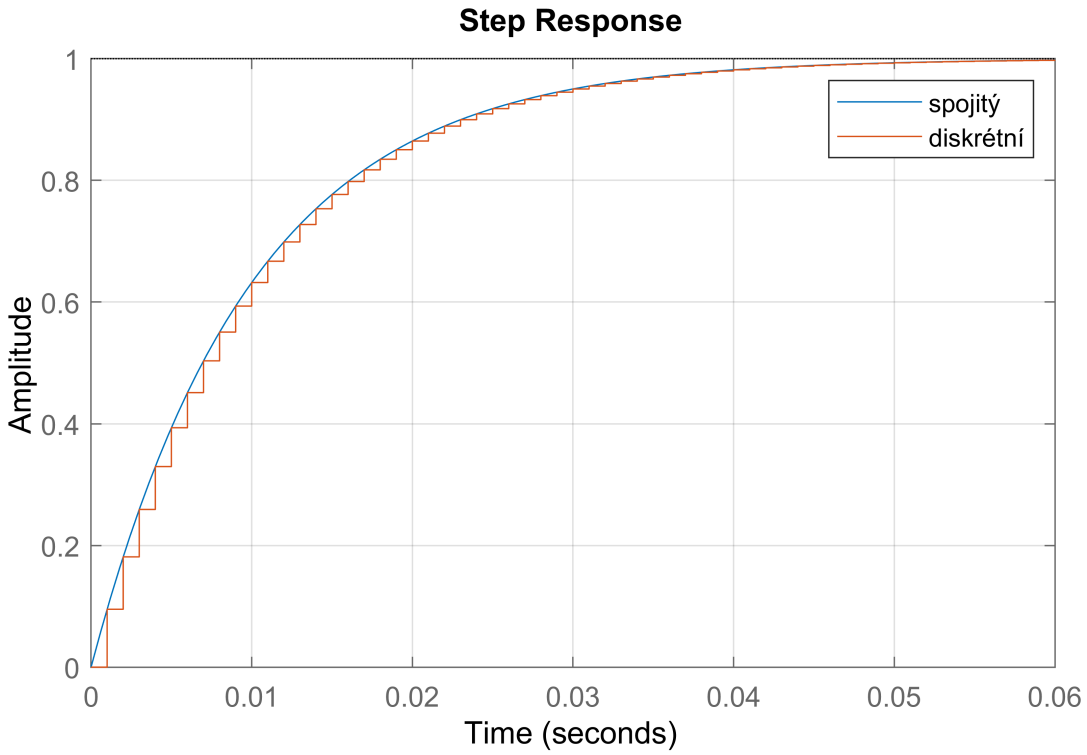
proto vynásobení výsledku počtem pólpárů motoru, tedy číslem tři. Tím je zajištěn převod na elektrický úhel. Nakonec je třeba přičíst počáteční úhel θ_0. Ten je však vzhledem k inicializační rutině popsané v kapitole 7.2 nulový, proto je možné tento krok za účelem zkrácení výpočetního času vynechat. Subsystém zajišťující popsaný postup je zobrazen na obrázku 7.6 a v příloze. [53]

Škálování výstupu

Výpočet hodnot pro registry PWM modulů zajišťuje blok SVGenDQ (*C2000 Space Vector Generator*). Jeho výpočty jsou oproti těm v kapitole 3.2 trochu odlišné. Vypočítané hodnoty se nachází v rozsahu $\langle -1; 1\rangle$. Ty je třeba nejdříve posunout do rozsahu $\langle 0; 2\rangle$. To lze udělat přičtením jedničky. Takto získané hodnoty však odpovídají nastavení PWM modulů uvedenému na obrázku 3.4 vlevo. V této práci je však využito nastavení uvedené na tomto obrázku vpravo. Generovaný vektor napětí by tak měl opačný směr. Pro správnou funkci je proto nutné výsledné hodnoty odečíst od maxima aktuálního rozsahu, tedy od dvou. Použité dvě matematické operace však lze sjednotit do jedné a ušetřit tak výpočetní čas. Ve výsledku tak stačí hodnoty generované blokem SVGenDQ odečíst od jedné. Vzniklý rozsah $\langle 0; 2\rangle$ je pak převeden na rozsah $\langle 0; \mathrm{TBPRD}\rangle$. Proměnné generované blokem SVGenDQ jsou před škálováním převedeny na datový typ *single*, aby nedošlo k přetečení. Subsystém zajišťující uvedené škálování je zobrazen na obrázku 7.7 a v příloze.

7.3.2 Rychlostní regulační smyčka

Při řízení motoru je nutné znát aktuální rychlost. V této práci je její výpočet prováděn pomocí rozdílu poloh za definovaný čas. Poloha je měřena vždy se stejným časovým odstupem (konstantní počet matematických operací prováděných procesorem) od začátku výpočtu proudové regulační smyčky. Vzhledem k tomu, že je proudová regulační smyčka spouštěna hardwarovým přerušením v pravidelných časových intervalech, lze takto změřené polohy využít k výpočtu rychlosti. Výpočet rozdílu poloh je prováděn v rámci proudové regulační smyčky. Od aktuální polohy se v každém časovém kroku



Obrázek 7.8: Porovnání přechodových charakteristik spojité a diskrétní verze rychlostního filtru

odečítá poloha změřená před 20 časovými kroky. Vypočítaný rozdíl je pak uložen do globální proměnné, odkud je při rychlostní regulaci přečtena aktuální hodnota. Výpočet rychlostní regulační smyčky je totiž vyvoláván na 20krát menší frekvenci než výpočet proudové regulační smyčky, pouze na 1 kHz.

Výpočet rychlosti z rozdílu poloh je proveden v jednom kroku společně se škálováním, a to vynásobením rychlosti škálovací konstantou, viz schéma v příloze. Jako maximální žádaná rychlost generovaná polohovým regulátorem je zvolena rychlost 3000 otáček za minutu. Tomu je také přizpůsobeno škálování, kde hodnota ±1 odpovídá rychlosti ±3000 RPM. Rychlostní regulátor však umí pracovat i s vyššími hodnotami.

Vypočítaná rychlost je dále filtrována. K filtraci byla použita přenosová funkce prvního řádu, dle rovnice 4.17, s časovou konstantou 0,01 s. Pomocí funkce *c2d* dostupné v toolboxu *Control System Toolbox* byla převedena do diskrétní podoby pro použitý časový krok 0,001 s. Porovnání přechodových charakteristik spojité a diskrétní verze je zobrazeno na obrázku 7.8.

Filtrovaná rychlost je následně zpracována rychlostním regulátorem. Ten jako požadovanou rychlost používá hodnotu vypočítanou polohovým regulátorem. Výstup z regulátoru v podobě požadovaného proudu i_q je omezen na hodnotu ±3/4,125 a uložen do globální proměnné. Schéma rychlostního regulátoru je opět uvedeno v příloze.

7.3.3 Polohová regulační smyčka

Nejvyšším stupněm při řízení polohy je polohová regulace. Ta v tomto případě běží na stejné frekvenci jako rychlostní regulační smyčka. Aktuální poloha je získána z posledního měření provedeného při proudové regulaci. Zde stačí provést odečtení inicializační hodnoty polohového čítače od aktuální hodnoty. Aktuální poloha společně s žádanou jsou zpracovány polohovým regulátorem. Tentokrát se však jedná již pouze o proporcionální regulátor, viz schéma v příloze.

Pro demonstraci chování motoru při řízení polohy byl zvolen sinusový průběh žádané hodnoty. Žádaná hodnota je generována tak, aby přímo odpovídala hodnotě čítače polohy enkodéru při dané poloze. Škálování je pak součástí proporcionální konstanty regulátoru. Výsledná žádaná rychlost generovaná tímto regulátorem je omezena na hodnoty z rozsahu $\langle -1; 1 \rangle$, aby mohla být přímo přivedena do rychlostního regulátoru.

Kapitola 8

Parametry regulátorů

8.1 Proudové regulátory

Na základě kapitoly 4.3 a parametrů použitého motoru byla zvolena konstanta proudového regulátoru

$$K_b = \frac{R}{L} = \frac{0,926\,7}{2,342 \cdot 10^{-4}} = 3\,957 \tag{8.1}$$

kde L je indukčnost vinutí statoru a R jeho elektrický odpor. Tato hodnota však není konečná. Aby byl zkrácen výpočetní čas, byl v regulátoru vynechán součinitel zohledňující časový krok výpočtu proudové regulační smyčky T_{proud}. Místo toho je tento součinitel zohledněn v rámci integrační konstanty regulátoru, respektive v rámci konstanty K_b. Výsledná konstanta regulátoru je tedy

$$K_{b_{scaled}} = \frac{R}{L} \cdot T_{proud} = 3\,957 \cdot \frac{1}{20\,000} = 0,198 \tag{8.2}$$

Druhou konstantu (K_a) je možné volit. Dle [54] je však doporučeno držet se přibližně v rozsahu

$$\frac{10 \cdot L}{\delta \cdot \tau} < K_a < \frac{\pi \cdot L}{5 \cdot T_{proud}} \tag{8.3}$$

kde δ je tlumicí faktor zavedený v kapitole 4.4 a τ je časová konstanta rychlostního filtru. Pokud je však nutné pohybovat se mimo tento rozsah, je doporučeno volit spíše menší hodnoty. Vyčíslením rovnice 8.3 je možné stanovit minimální a maximální hodnotu

$$K_{a\,min} = \frac{10 \cdot L}{\delta \cdot \tau} = \frac{10 \cdot 2,342 \cdot 10^{-4}}{4 \cdot 0,01} = 0,058\,55 \tag{8.4a}$$

$$K_{a\,max} = \frac{\pi \cdot L}{5 \cdot T_{proud}} = \frac{\pi \cdot 2,342 \cdot 10^{-4}}{5 \cdot \frac{1}{20\,000}} = 2,943 \tag{8.4b}$$

Tlumicí faktor δ je zde zvolen v souvislosti s rychlostní regulační smyčkou.

V matematickém modelu, na jehož základě byla volba parametrů odvozena, vstupuje do regulátoru hodnota protékajícího proudu v ampérech a výstupem je přímo hodnota napětí ve voltech. Proto je nutné konstantu K_a naškálovat s ohledem na škálování vstupních a výstupních hodnot. [55]

Velikost této konstanty ovlivňují následující body:

- Vzhledem k principu měření a škálování proudů a následnému zmenšení vektoru proudu Clarkeové transformací odpovídá rozsah $\langle -1; 1 \rangle$ hodnot vstupujících do regulátoru rozsahu proudů $\langle -4,125; 4,125 \rangle$ A. Konstanta K_a proto musí být vynásobena součinitelem 4,125.
- SVM modulace zmenší vektor napětí na $\frac{\sqrt{3}}{2}$ původní hodnoty. Konstanta K_a proto musí být vynásobena součinitelem $\frac{2}{\sqrt{3}}$.
- Vektor napětí (po zmenšení SVM modulací) o velikosti 1 odpovídá 24 V. Konstanta K_a proto musí být vynásobena součinitelem $\frac{1}{24}$.

Výsledné hranice škálované konstanty K_a jsou tedy

$$K_{a\,min\,scaled} = K_{a\,min} \cdot 4,125 \cdot \frac{2}{\sqrt{3}} \cdot \frac{1}{24} = 0,058\,55 \cdot \frac{11\sqrt{3}}{96} = 0,011\,62 \tag{8.5a}$$

$$K_{a\,max\,scaled} = K_{a\,max} \cdot 4,125 \cdot \frac{2}{\sqrt{3}} \cdot \frac{1}{24} = 2,943 \cdot \frac{11\sqrt{3}}{96} = 0,584 \tag{8.5b}$$

Výsledná hodnota konstanty

$$K_{a\,scaled} = 0,05 \tag{8.6}$$

byla zvolena při ladění rychlostního regulátoru.

8.2 Rychlostní regulátor

Při návrhu rychlostního regulátoru je nejdříve nutné znát parametry motoru a zátěže. Řízení motoru je v této práci navrhováno pro nezatížený motor, celkový moment setrvačnosti J je proto roven momentu setrvačnosti motoru.

Dále je nutné znát magnetický indukční tok rotoru ψ. Ten je při použití SI jednotek shodný s konstantou zpětného elektromotorického napětí, kterou lze jednoduše určit analýzou průběhu napětí vznikajícího mezi dvěma fázemi při roztočení motoru na konstantní úhlovou rychlost externím motorem, viz obr. 8.1. Požadovanou jednotkou jsou volty (špička, fáze proti nule) na elektrický radián za sekundu. Na základě hodnot odečtených z osciloskopu je proto možné spočítat magnetický indukční tok vztahem

$$\psi = \frac{1,06}{35,21} \cdot \frac{1}{\sqrt{3}} \cdot \frac{1}{2\,\pi} = 2,766 \cdot 10^{-3}\ \frac{\mathrm{V}}{\frac{\mathrm{rad_{el}}}{\mathrm{s}}} \tag{8.7}$$

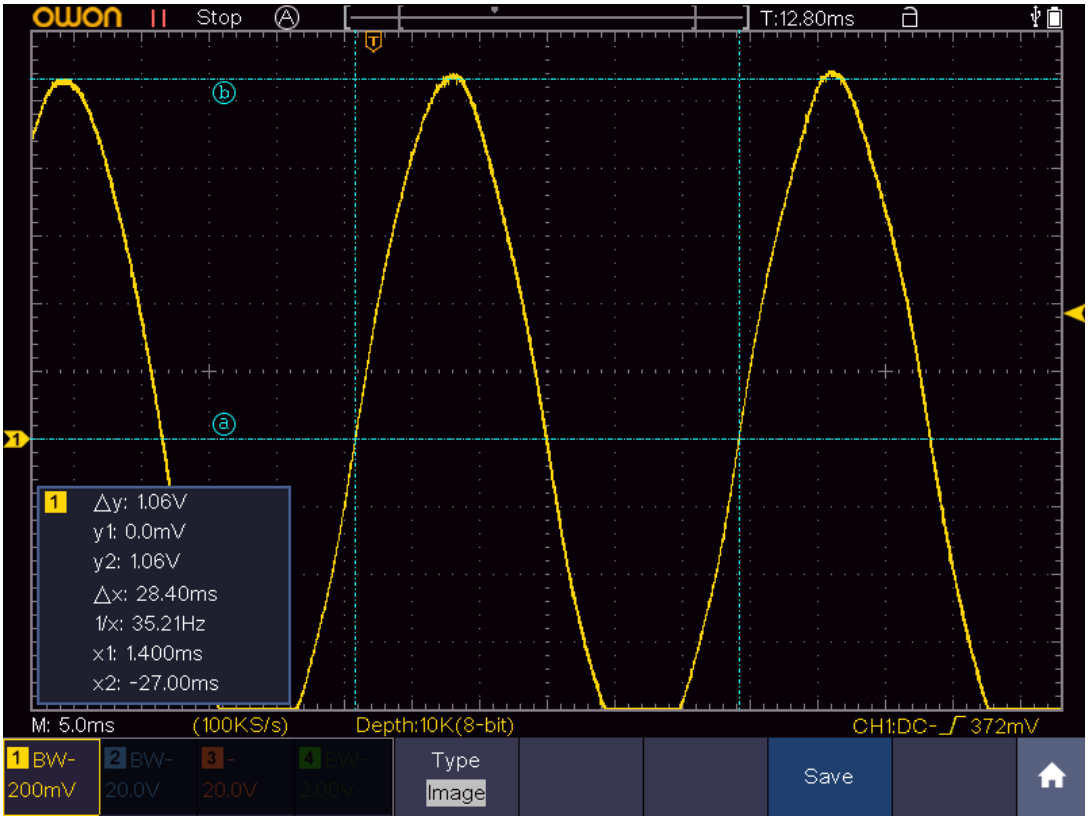
kde součinitel $\frac{1}{\sqrt{3}}$ zajišťuje převod ze sdruženého na fázové napětí a součinitel $\frac{1}{2\,\pi}$ zajišťuje převod z elektrických Hz na elektrické $\frac{\mathrm{rad}}{\mathrm{s}}$. Bylo provedeno několik měření při různých úhlových rychlostech, hodnoty se mírně zvyšují s rostoucí úhlovou rychlostí. [3] [39] [56]

Nyní už je možné vypočítat konstantu

$$K = \frac{P\,\psi}{2\,J} = \frac{6 \cdot 2,766 \cdot 10^{-3}}{2 \cdot 3,54 \cdot 10^{-7}} = 23\,441\ \frac{\mathrm{V \cdot s}}{\mathrm{rad \cdot kg \cdot m^2}} \tag{8.8}$$

a s její pomocí parametr

$$K_c = \frac{1}{\delta\,K\,\tau} = \frac{1}{4 \cdot 23\,441 \cdot 0,01} = 1,067 \cdot 10^{-3} \tag{8.9}$$



Obrázek 8.1: Měření zpětného elektromotorického napětí

Z principu vyplývá, že proud i_q^{ref} vystupující z rychlostního regulátoru musí být před vstupem do proudového regulátoru škálován stejným způsobem jako měřený proud i_q. Toho lze dosáhnout škálováním konstanty K_c. Tu je proto nutné vydělit velikostí proudu 4,125 A. Dále je nutné zohlednit škálování rychlostí vstupujících do rychlostního regulátoru. V kapitole 7.3.2 bylo zvoleno, že rozsah $\langle -1; 1\rangle$ škálovaných hodnot odpovídá rozsahu $\langle -3000; 3000\rangle$ otáček za minutu. Matematický model však předpokládá úhlovou rychlost v (mechanických) radiánech za sekundu. Výslednou konstantu $K_{c\,scaled}$ tak lze vypočítat ze vztahu

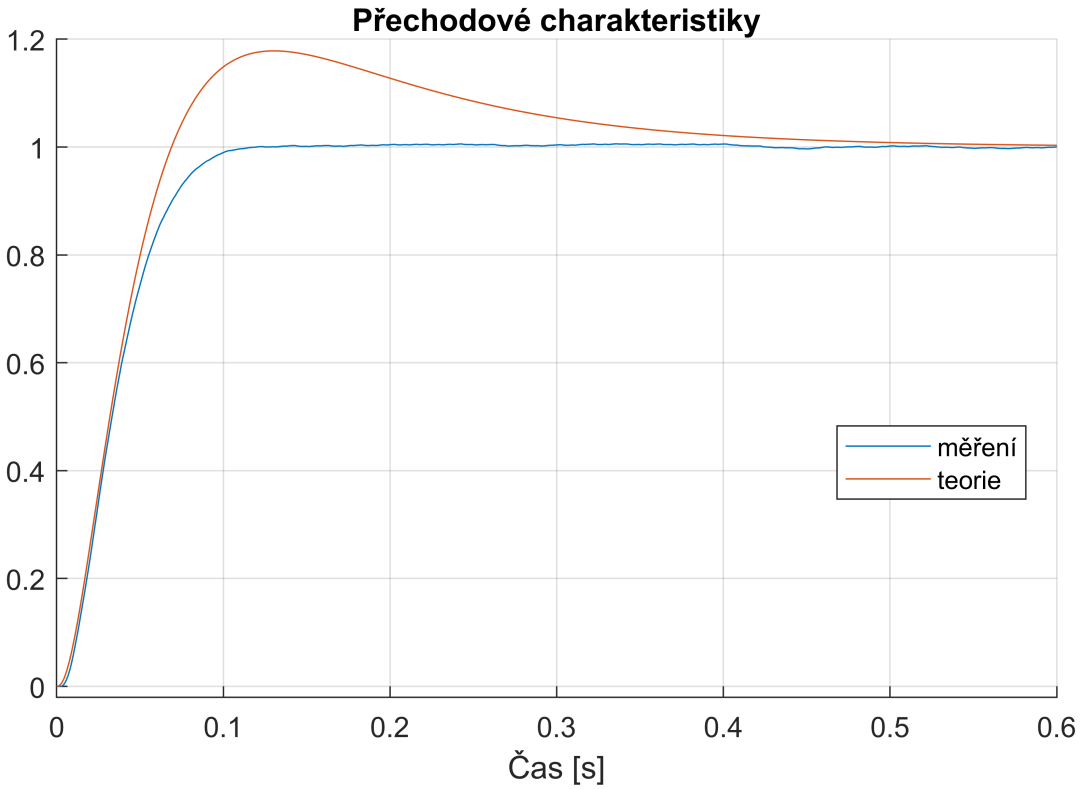
$$K_{c\,scaled} = K_c \cdot \frac{\frac{3000}{60} \cdot 2\,\pi}{4,125} = 1,067 \cdot 10^{-3} \cdot \frac{100\,\pi}{4,125} = 0,081\,3 \tag{8.10}$$

Zbývající konstantu K_d je možné určit ze vztahu

$$K_d = \frac{1}{\delta^2\,\tau} = \frac{1}{4^2 \cdot 0,01} = 6,25 \tag{8.11}$$

Její škálování je analogicky stejné jako škálování konstanty K_b proudových regulátorů. Tentokrát je však použit časový krok rychlostní regulační smyčky $T_{rychlost}$ a výsledná škálovaná konstanta je

$$K_{d\,scaled} = K_d \cdot T_{rychlost} = 6,25 \cdot \frac{1}{1000} = 6,25 \cdot 10^{-3} \tag{8.12}$$



Obrázek 8.2: Porovnání přechodových charakteristik rychlostní regulace

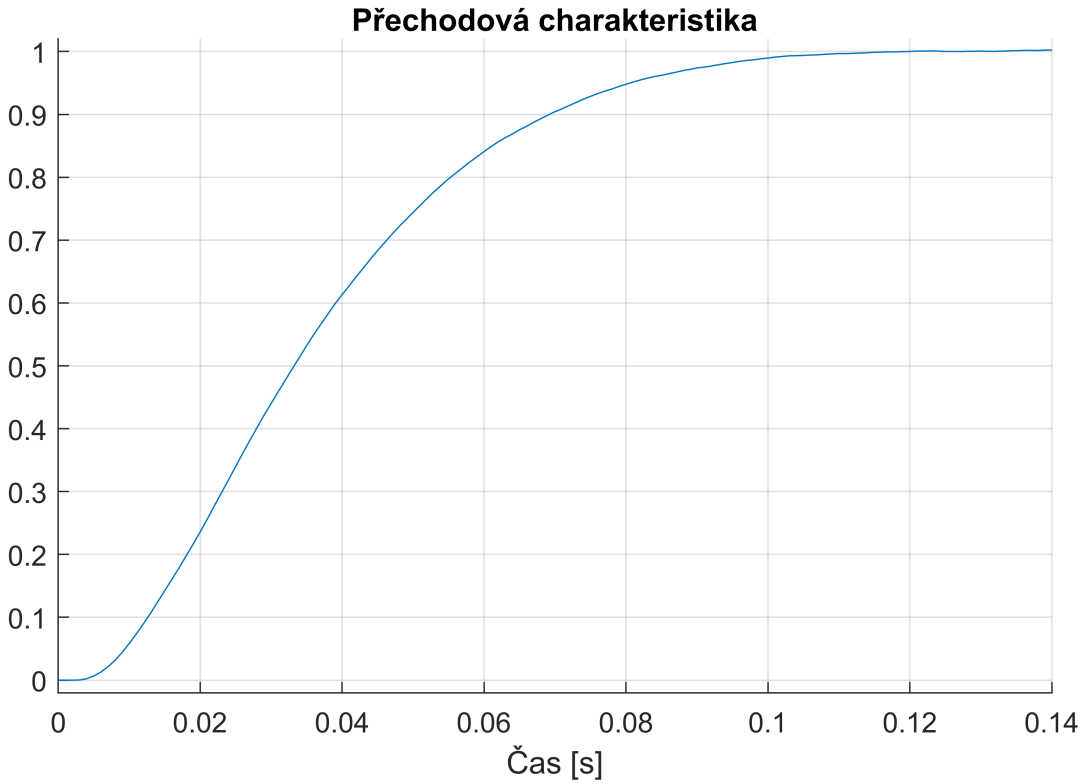
[55]

Použitý matematický model není úplně přesný, stejně tak jeho parametry. Navíc je vytvořen jako spojitý, přičemž skutečná regulace probíhá diskrétně a dokonce s různými časovými kroky. Následkem těchto problémů neodpovídá naměřená přechodová charakteristika s dostatečnou přesností té teoretické, jak je patrné z obrázku 8.2. Proto je zbytečné dále ladit parametry pomocí tlumicího faktoru δ. Další ladění parametrů je možné provést metodou pokus-omyl změnou parametru K_a a případně parametrů K_c a K_d. Při volbě K_a dle rovnice 8.6 je však chování motoru vyhovující (viz obr. 8.3), proto již není nutné parametry K_c a K_d dále ladit. Pro upřesnění je vhodné ještě dodat, že vypočítané parametry nevytvoří takto pěknou přechodovou charakteristiku při všech možných hodnotách parametru K_a. Například při jeho volbě na spodní hranici (dle 8.5a) dojde k relativně velkému překmitu a výraznému prodloužení času potřebného pro dosažení ustáleného stavu.

Přechodová charakteristika byla měřena při skokové změně žádané hodnoty z 0 na 3000 ot/min.

8.3 Polohový regulátor

Ladění polohového regulátoru bylo provedeno metodou pokus-omyl. Vzhledem k tomu, že byl polohový regulátor zvolen pouze jako proporcionální (bez integrační složky), stačí naladit jen jeden parametr. To metodu pokus-omyl výrazně zjednodušuje. Nakonec

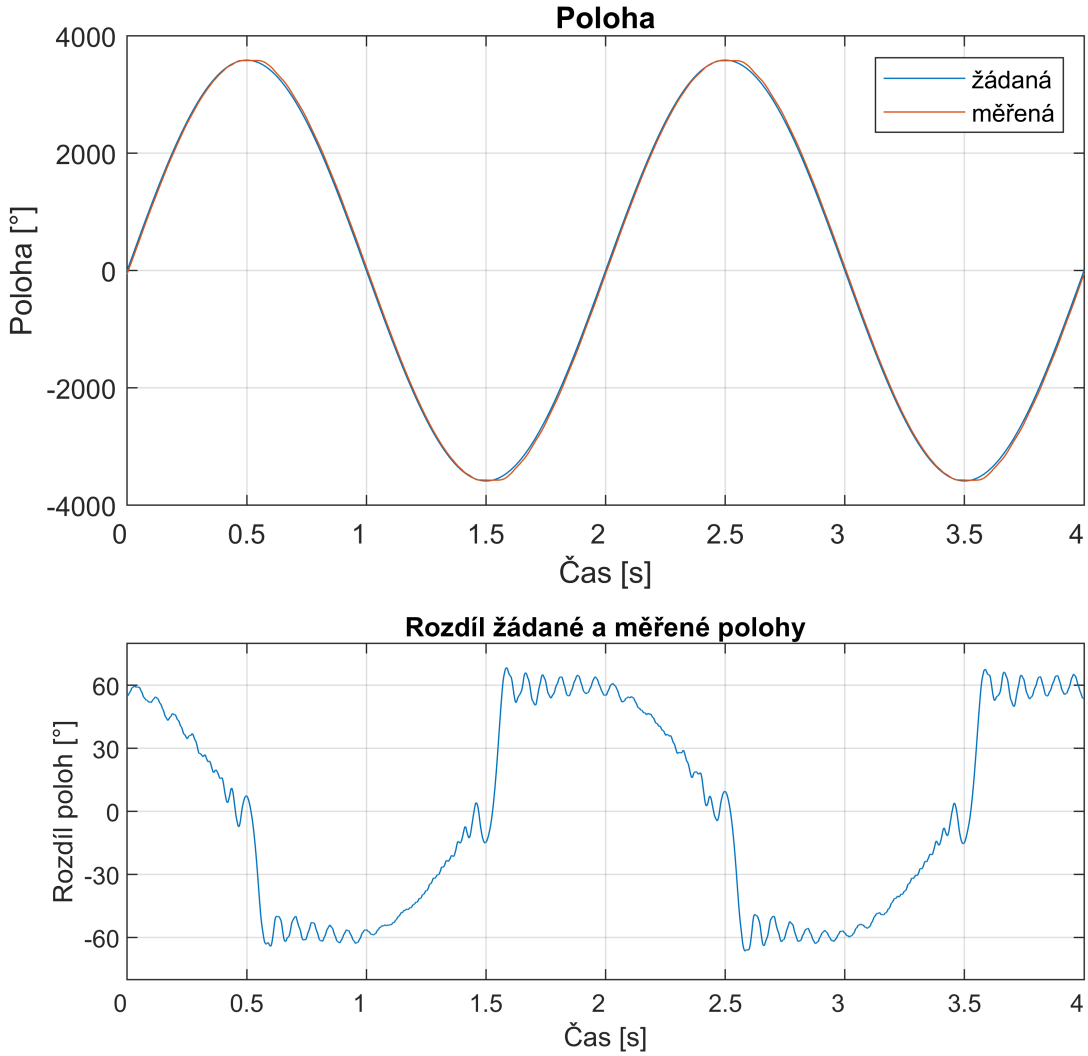


Obrázek 8.3: Přechodová charakteristika rychlostní regulace pro finální parametry

byla zvolena hodnota proporcionální složky

$$K_p = 5 \cdot 10^{-4} \tag{8.13}$$

Na obrázku 8.4 je zobrazeno chování motoru při sinusovém průběhu žádané polohy. Motor má celkově problémy s nízkými rychlostmi, což se negativně projevuje na řízení polohy.



Obrázek 8.4: Polohová regulace při sinusovém průběhu žádané hodnoty

Kapitola 9
Doplňující informace a zhodnocení

Použití Simulinku při návrhu řídicího algoritmu se ukázalo v některých ohledech velice výhodné. Hlavní výhodou byla přehlednost algoritmu díky použití blokových schémat. Dále však byly užitečné i již připravené bloky různých operací (použité transformace souřadných systémů, generátor vektorové modulace, diskrétní přenosová funkce použitá jako filtr rychlosti apod.), předpřipravené nastavení hardwaru či možnost simulace vytvořených schémat. MATLAB, ve kterém Simulik běží, je navíc také velmi silný nástroj v oblasti zpracování naměřených dat. Možnost ladění parametrů modelu při spuštěném řízení byla nepostradatelná při ladění regulátorů a kontrole funkčnosti algoritmu. Objevily se však také různé komplikace, viz například kapitoly 9.1 a 9.2.

Co se týče použitého hardwaru od Texas Instruments, nevyskytl se kromě špatné konfigurace EEPROM (viz kapitola 9.1) žádný významnější problém. Také dokumentace k němu byla velmi podrobná. Dobrá dokumentace byla k dispozici také k většině funkcí MATLABu a Simulinku. Problém s kvalitou dokumentace nastával pouze u některých bloků dostupných prostřednictvím balíčku hardwarové podpory. Konkrétně například u bloku pro generaci vektorové modulace. Zde nebylo popsáno, jakým způsobem jsou výsledné hodnoty počítány a nebylo tedy zřejmé, jakým způsobem je následně škálovat. Vzhledem k možnosti simulace to však nebyl problém zjistit. Za nevýhodu je možné považovat, že programy MATLAB a Simulink včetně veškeré dokumentace jsou celé v anglickém jazyce, stejně tak jako dokumentace k použitému hardwaru.

9.1 Sériová komunikace

Prvním krokem při návrhu řídicího algoritmu bylo zprovoznění balíčku hardwarové podpory v prostředí Simulink. Asi největším problémem zde byla sériová komunikace mezi Simulinkem a vývojovou deskou. V první řadě bylo nutné přeprogramovat paměť EEPROM na vývojové desce, neboť byla z výroby špatně nakonfigurovaná a nepodporovala komunikaci přes virtuální COM port, viz [57]. Dále bylo nutné zvolit jeden ze tří způsobů komunikace – *serial*, *serial(using xcp)* a *CAN*. Výchozí *serial* umožňuje za chodu ladění všech typů parametrů modelu. Jeho nevýhodou je, že v Simulinku neukládá data z celého měření, ale jen pár posledních vzorků. Je sice možné tyto sady vzorků archivovat do souborů, tím ale vznikají tisíce souborů, se kterými se velmi špatně pracuje. Při použití druhé varianty, přes XCP protokol, tento problém odpadá. Je však nahrazen jinými. Pro úspěšnou komunikaci použitého modelu s vývojovou deskou přes XCP protokol tak bylo změněno výchozí chování všech parametrů z laditelných

(*tunable*) na neladitelné (*inlined*). Tento krok navíc zvýšil efektivitu vygenerovaného kódu. Parametry, které bylo zapotřebí ladit za provozu (parametry regulátorů, žádané hodnoty proudu, rychlosti a polohy, parametry pro ovládání modelu apod.), musely pak být nastaveny na laditelné samostatně. [58] [59] [60] [61]

Dalším problémem souvisejícím se sériovou komunikací byla nemožnost zobrazení a ukládání dat na frekvenci 20 kHz, na které běží proudová regulační smyčka. Všechna měření uvedená v této práci jsou proto provedena na frekvenci 1 kHz, na které běží ostatní regulační smyčky.

9.2 Omezení vyplývající z použití Simulinku

Možnost využití Simulinku pro tvorbu programů pro procesory C2000™ je dána existencí balíčku hardwarové podpory. Ten má však různá omezení. Jedním z nich je to, že pro práci s některými moduly (ePWM, ADC, eQEP apod.) je dostupný vždy pouze jeden univerzální blok. Ten zajistí počáteční konfiguraci modulu i následnou interakci s jeho registry v průběhu řízení. Takovýto blok však není možné použít pro jeden modul (např. ePWM4 nebo eQEP1) vícekrát. Pokud je tedy zapotřebí přistupovat k registrům daného modulu v různých částech modelu pracujících s různým časovým krokem, je možné blok buď použít pouze v jedné z nich a následně sdílet hodnotu s ostatními (jako je tomu v této práci u hodnoty čítače polohy QPOSCNT), nebo blok umístit do samostatného subsystému a ten vyvolávat z obou částí modelu. V první variantě však hodnota při použití v dalších částech modelu již nemusí být dostatečně aktuální. Ve druhé variantě je zase omezení, že pokud je subsystém vyvoláván z části modelu vyvolávané asynchronně (například hardwarovým přerušením), není ho možné vyvolávat zároveň z jiné části modelu pracující s pravidelným časovým krokem (v této práci například polohovou nebo rychlostní regulační smyčkou a zároveň asynchronní proudovou regulační smyčkou). Dále také může být v různých částech modelu požadován přístup k různým registrům tohoto modulu (v jedné například resetování čítače polohy, ve druhé hodnota čítače polohy a ve třetí aktuální směr). V takovém případě jsou však při použití druhé varianty provedeny všechny operace (zápis registru pro resetování čítače a čtení obou registrů pro získání zmíněných hodnot), což zbytečně prodlužuje výpočetní čas, a je proto při řízení v reálném čase nežádoucí. Při tvorbě řídicího algoritmu přímo v C kódu (bez použití Simulinku) je přitom každá z těchto operací zastoupena jedním jednoduchým příkazem, který je možné použít samostatně kdekoli a opakovaně. Bylo by sice možné v Simulinku bloky zajišťující takovéto chování doprogramovat, to by však vyžadovalo výrazně hlubší znalosti.

Vývoj řídicích algoritmů v Simulinku je také v některých ohledech rizikovější z hlediska chyb. Jednak můžou použité bloky obsahovat chyby (obvzláště ty v balíčku hardwarové podpory, neboť jsou méně používané) a také musí být model přeložen do C kódu, přičemž kompilátor může také obsahovat chyby. Na druhou stranu dokáže Simulink odhalit spoustu chyb zaviněných uživatelem.

Kapitola 10

Závěr

Cílem této práce bylo zprovoznit řízení motoru Yaskawa SGMM-A1S312 vektorovou metodou. Ta je založena na souboru matematických operací zjednodušujících regulaci proudů v motoru, které přímo souvisejí s krouticím momentem motoru.

V první části práce byl popsán princip vektorového řízení, princip realizace vypočítaného akčního zásahu pomocí vektorové modulace a postup návrhu regulátorů. Ve druhé části pak byla popsána realizace uvedených principů při řízení skutečného motoru. Velký důraz přitom byl kladen na správné provedení měření a jeho vyhodnocení. Řídicí algoritmus byl navržen pomocí blokových schémat v programu Simulink.

K řízení motoru byla využita vývojová deska LaunchPad F28069M s výkonovým modulem BOOSTXL-DRV8301. Řízení bylo navrženo pro nezatížený motor a podařilo se ho zprovoznit na úrovni polohové regulace. Cíle práce tak bylo dosaženo.

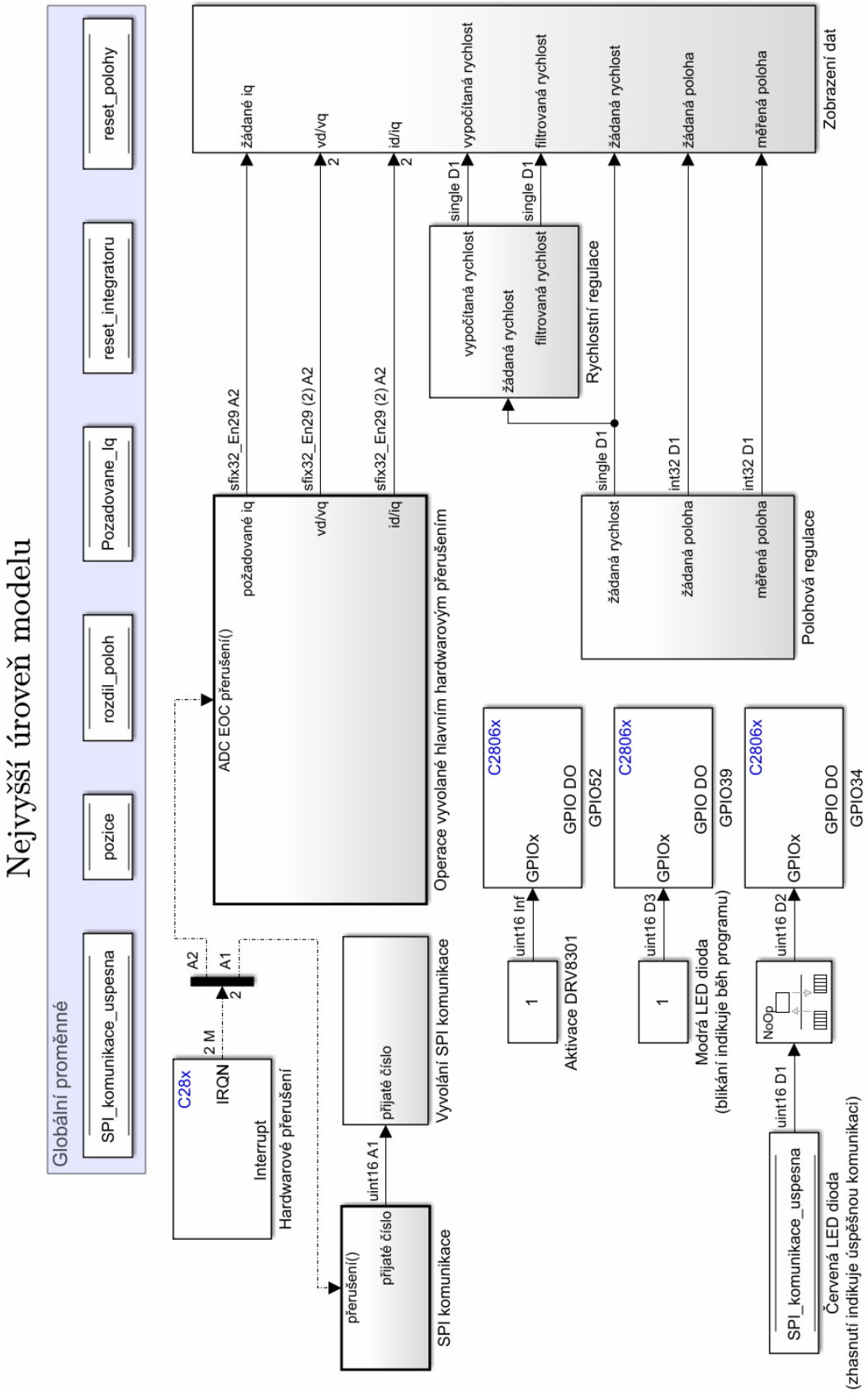
Přílohy

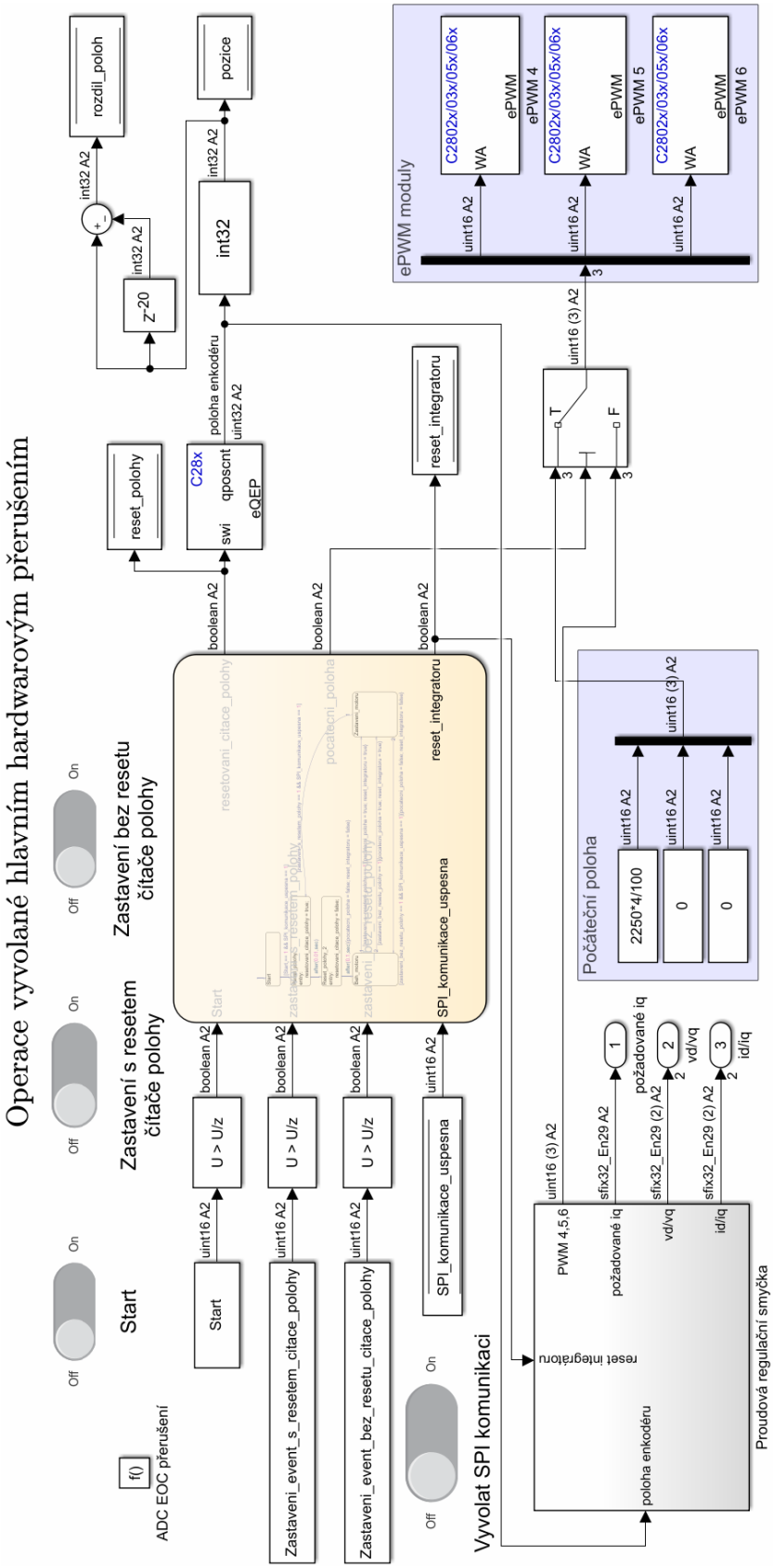
Příloha A

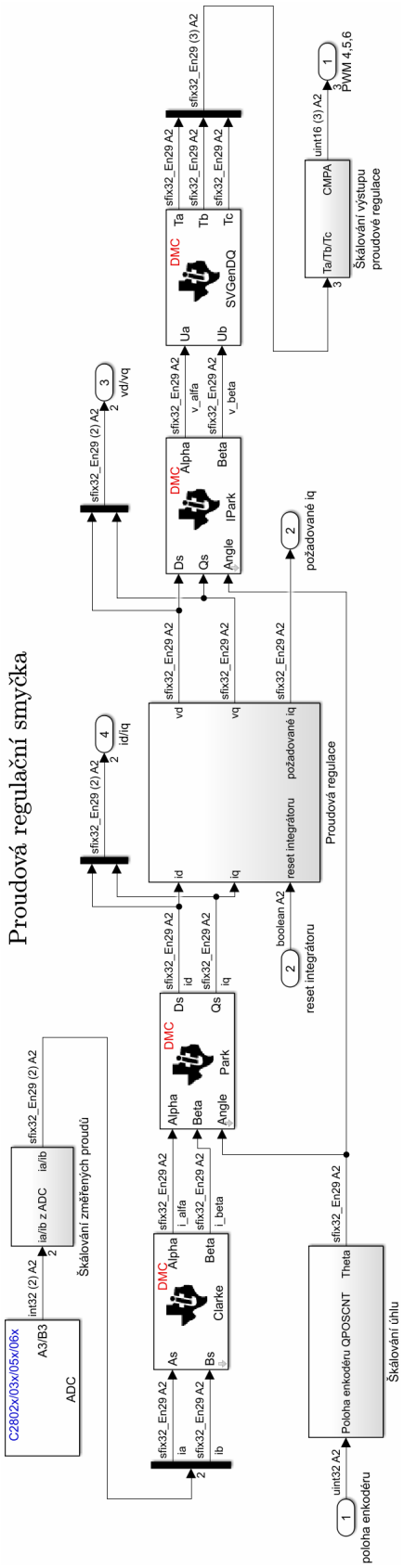
Schémata modelu

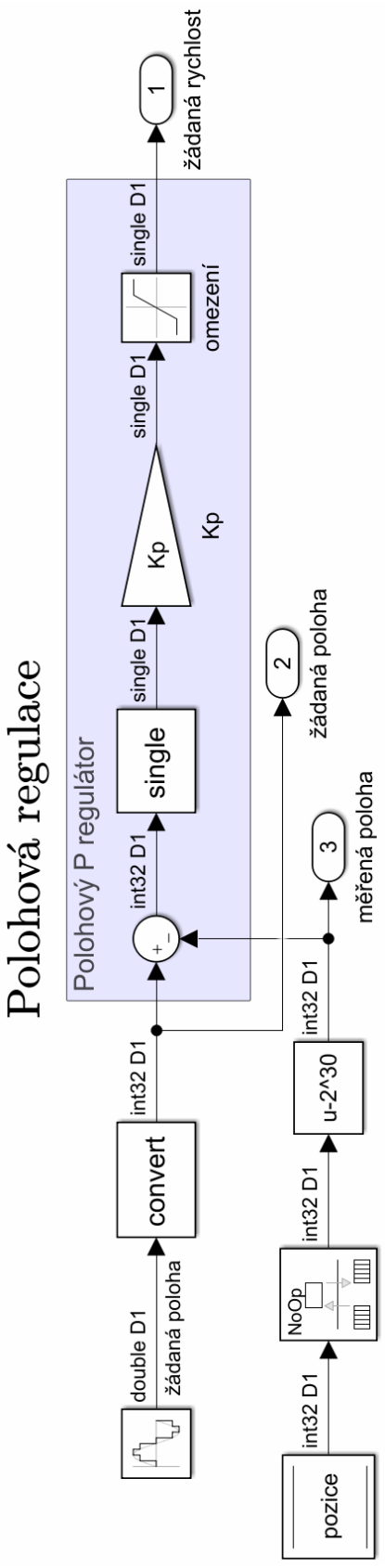
Hierarchie modelu

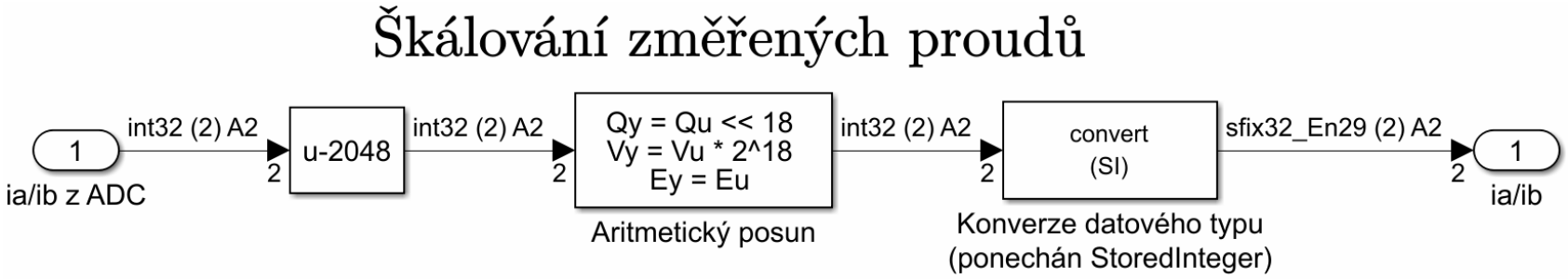
```
BP_model.slx
└── Nejvyšší úroveň modelu
    ├── Operace vyvolané hlavním hardwarovým přerušením
    │   └── Proudová regulační smyčka
    │       ├── Škálování změřených proudů
    │       ├── Škálování úhlu
    │       ├── Proudová regulace
    │       │   └── Proudový PI regulátor
    │       └── Škálování výstupu proudové regulace
    ├── Polohová regulace
    ├── Rychlostní regulace
    │   └── Rychlostní PI regulátor
    ├── Vyvolání SPI komunikace
    ├── SPI komunikace
    └── Zobrazení dat
```

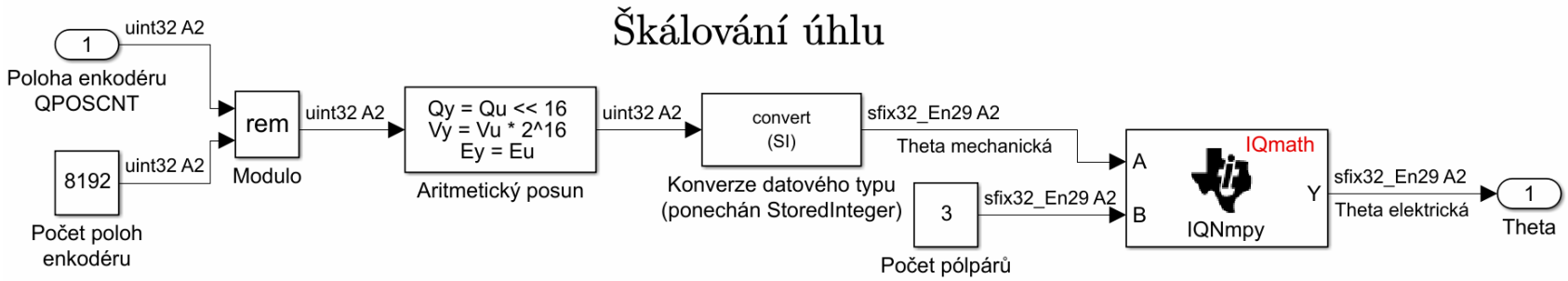

Nejvyšší úroveň modelu
Globální proměnné
SPI_komunikace_uspesna
pozice
rozdil_poloh
Pozadovane_Iq
reset_integratoru
reset_polohy
C28x
IRQN
Interrupt
Hardwarové přerušení
2 M
A2
A1
2
přerušení()
přijaté číslo
SPI komunikace
uint16 A1
přijaté číslo
Vyvolání SPI komunikace
ADC EOC přerušení()
požadované iq
vd/vq
id/iq
Operace vyvolané hlavním hardwarovým přerušením
sfix32_En29 A2
sfix32_En29 (2) A2
sfix32_En29 (2) A2
2
2
žádané iq
vd/vq
id/iq
Zobrazení dat
single D1
single D1
vypočítaná rychlost
filtrovaná rychlost
vypočítaná rychlost
žádaná rychlost
filtrovaná rychlost
Rychlostní regulace
single D1
int32 D1
int32 D1
žádaná rychlost
žádaná poloha
měřená poloha
Polohová regulace
žádaná rychlost
žádaná poloha
měřená poloha
C2806x
GPIOx
GPIO DO
GPIO52
uint16 Inf
1
Aktivace DRV8301
C2806x
GPIOx
GPIO DO
GPIO39
uint16 D3
1
Modrá LED dioda
(blikání indikuje běh programu)
C2806x
GPIOx
GPIO DO
GPIO34
uint16 D2
NoOp
uint16 D1
SPI_komunikace_uspesna
Červená LED dioda
(zhasnutí indikuje úspěšnou komunikaci)

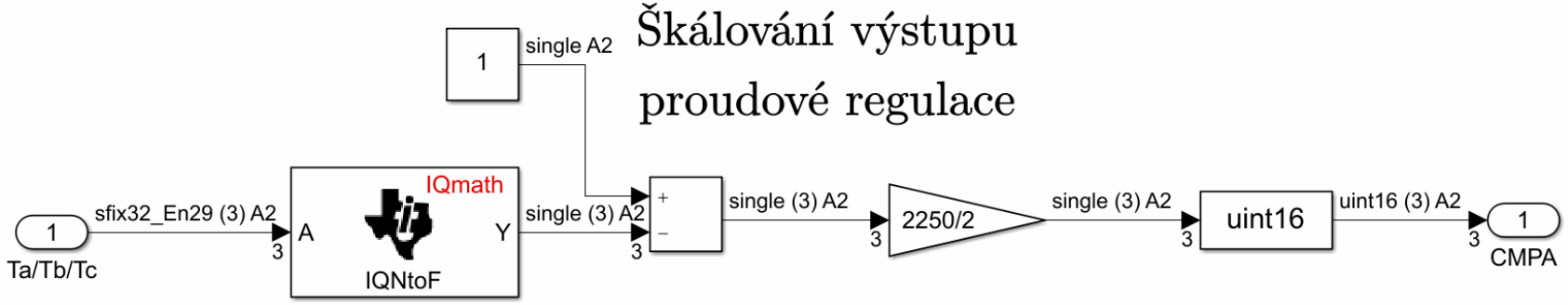

Operace vyvolané hlavním hardwarovým přerušením
ADC EOC přerušení
Start
Zastavení s resetem čítače polohy
Zastavení bez resetu čítače polohy
Vyvolat SPI komunikaci
Zastaveni_event_s_resetem_citace_polohy
Zastaveni_event_bez_resetu_citace_polohy
SPI_komunikace_uspesna
U > U/z
reset_polohy
reset_integratoru
rozdil_poloh
pozice
poloha enkodéru
ePWM moduly
C2802x/03x/05x/06x
ePWM 4
ePWM 5
ePWM 6
Počáteční poloha
2250*4/100
Proudová regulační smyčka
PWM 4,5,6
požadované iq
vd/vq
id/iq

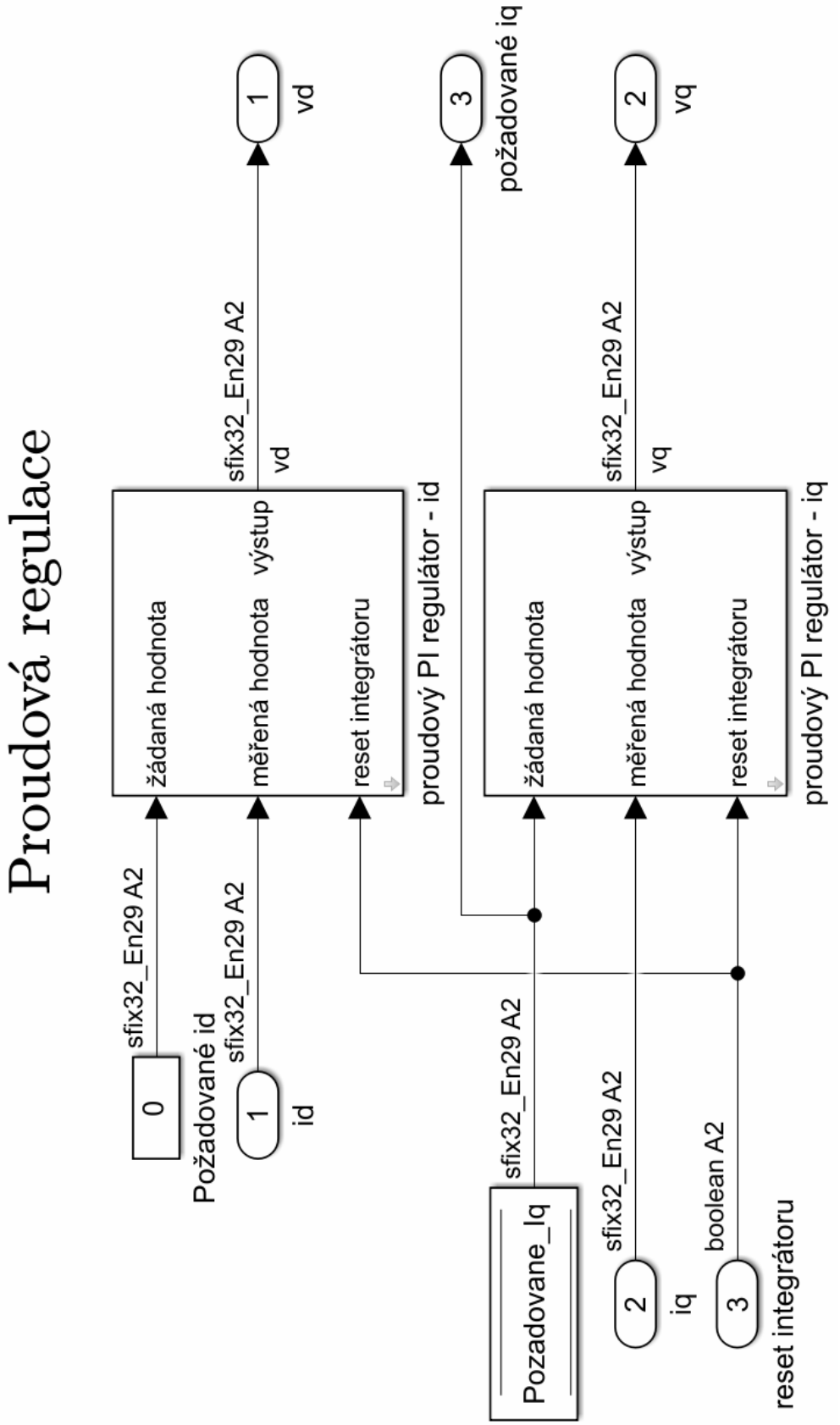

Proudová regulační smyčka
C2802x/03x/05x/06x
A3/B3
ADC
int32 (2) A2
ia/ib z ADC
sfix32_En29 (2) A2
ia/ib
Škálování změřených proudů
DMC
Alpha
Beta
Clarke
As
Bs
sfix32_En29 A2
ia
ib
i_alfa
i_beta
Ds
Qs
Park
Angle
id
iq
sfix32_En29 (2) A2
id/iq
vd
vq
Proudová regulace
reset integrátoru
požadované iq
boolean A2
uint32 A2
Poloha enkodéru QPOSCNT
Theta
Škálování úhlu
poloha enkodéru
IPark
vd/vq
v_alfa
v_beta
SVGenDQ
Ua
Ub
Ta
Tb
Tc
Ta/Tb/Tc
CMPA
uint16 (3) A2
PWM 4,5,6
Škálování výstupu proudové regulace

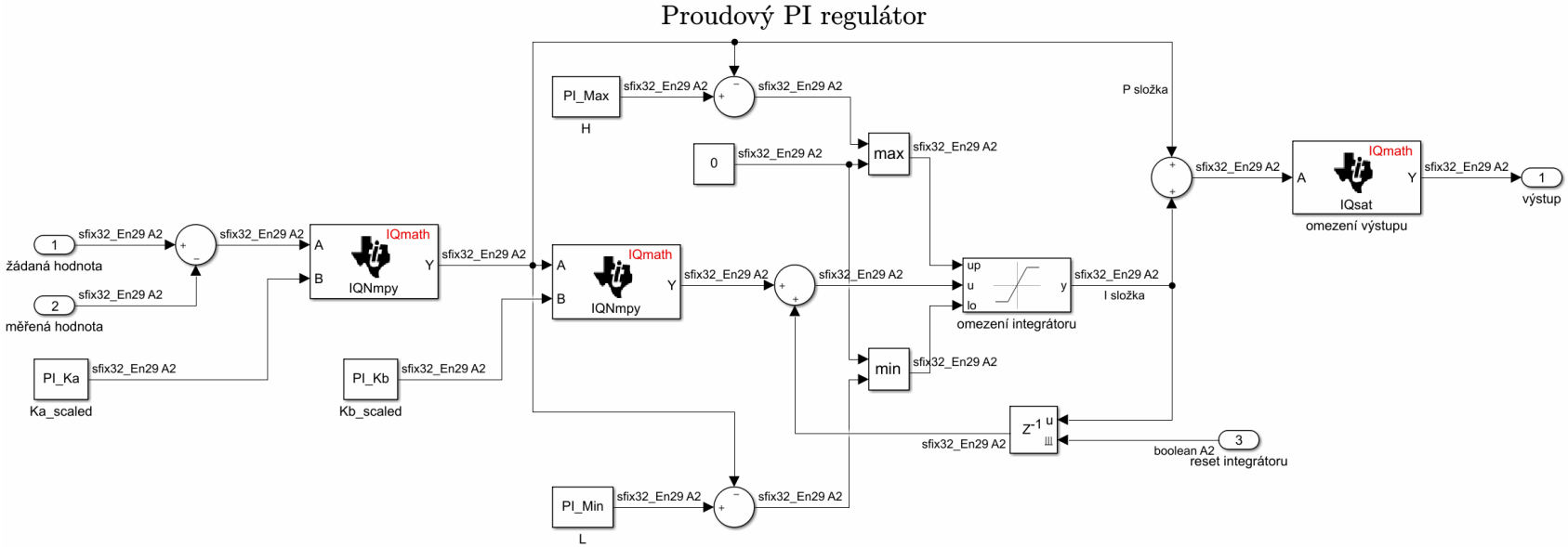

Polohová regulace
Polohový P regulátor
double D1
žádaná poloha
convert
int32 D1
int32 D1
single
single D1
Kp
Kp
single D1
omezení
single D1
žádaná rychlost
pozice
int32 D1
NoOp
int32 D1
u-2^30
int32 D1
měřená poloha
žádaná poloha

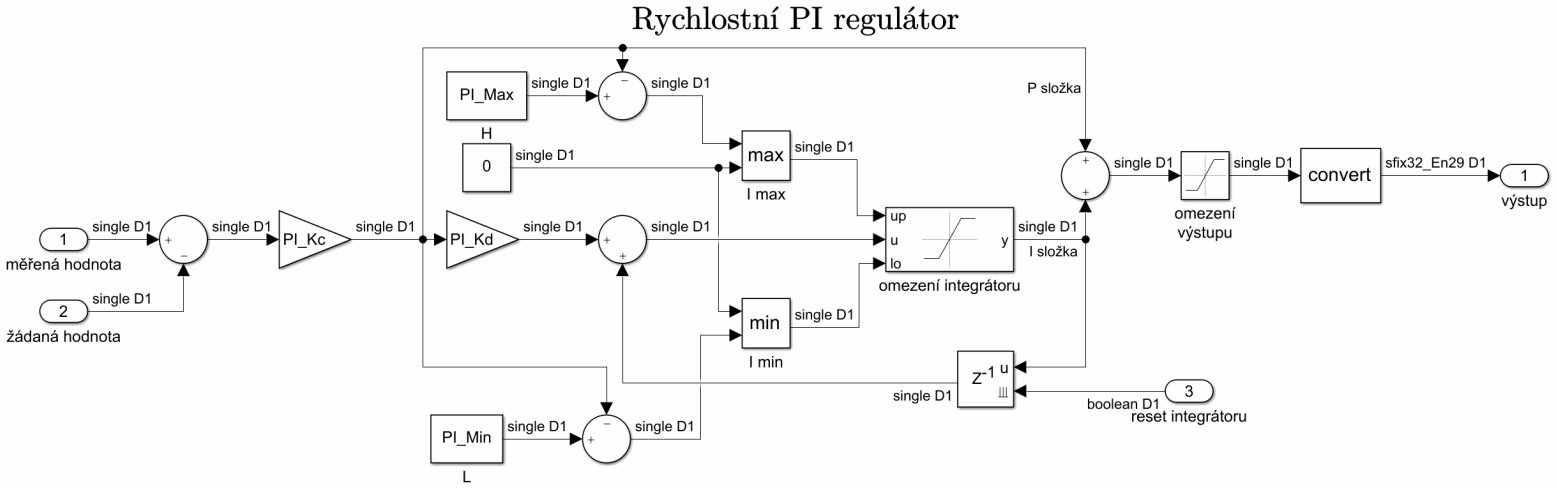

Škálování změřených proudů
1
ia/ib z ADC
int32 (2) A2
u-2048
int32 (2) A2
Qy = Qu << 18
Vy = Vu * 2^18
Ey = Eu
Aritmetický posun
int32 (2) A2
convert
(SI)
Konverze datového typu
(ponechán StoredInteger)
sfix32_En29 (2) A2
1
ia/ib

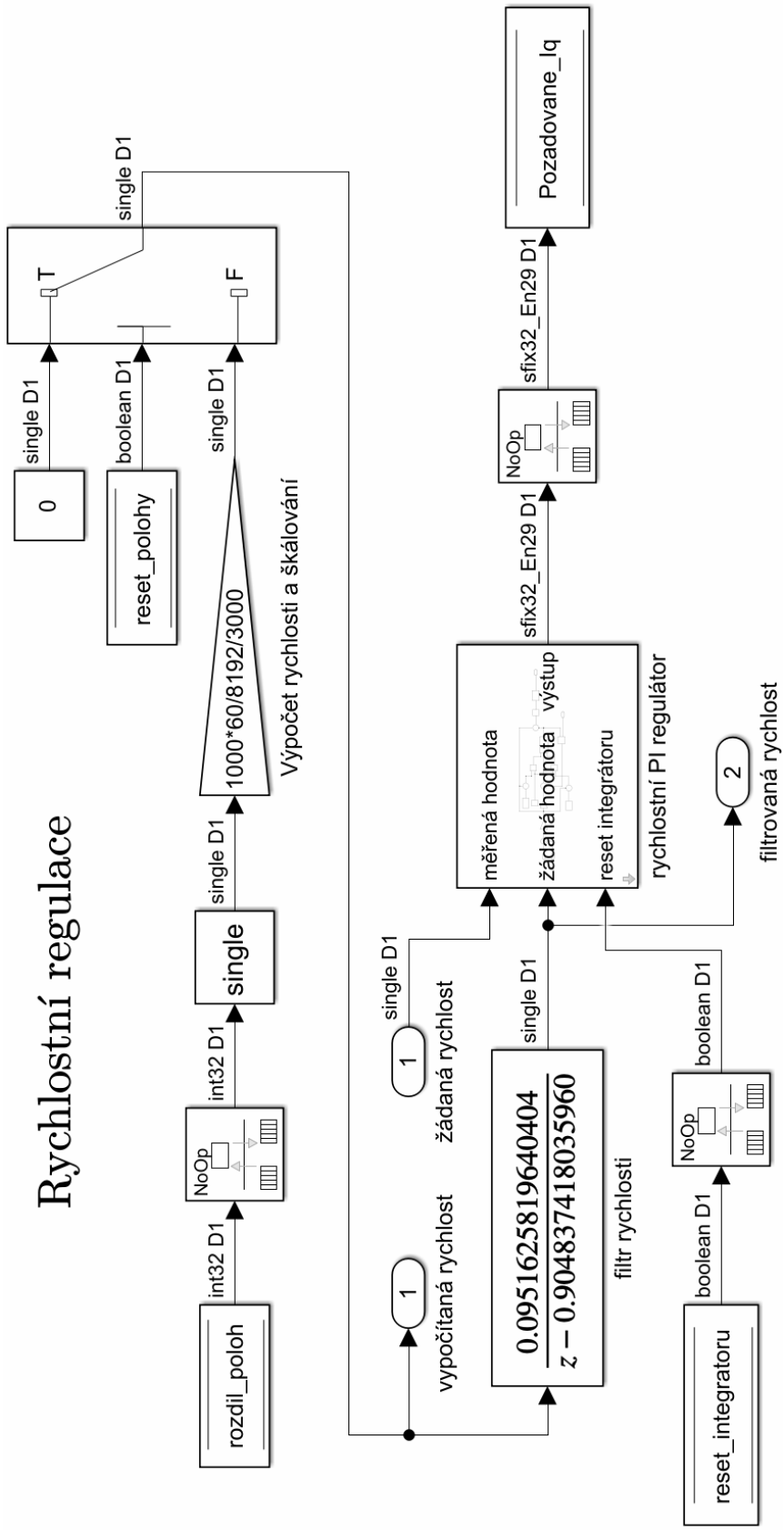

Škálování úhlu
1
Poloha enkodéru
QPOSCNT
uint32 A2
8192
Počet poloh
enkodéru
rem
Modulo
uint32 A2
Qy = Qu << 16
Vy = Vu * 2^16
Ey = Eu
Aritmetický posun
uint32 A2
convert
(SI)
Konverze datového typu
(ponechán StoredInteger)
sfix32_En29 A2
Theta mechanická
3
Počet pólpárů
sfix32_En29 A2
A
B
IQmath
IQNmpy
Y
sfix32_En29 A2
Theta elektrická
1
Theta

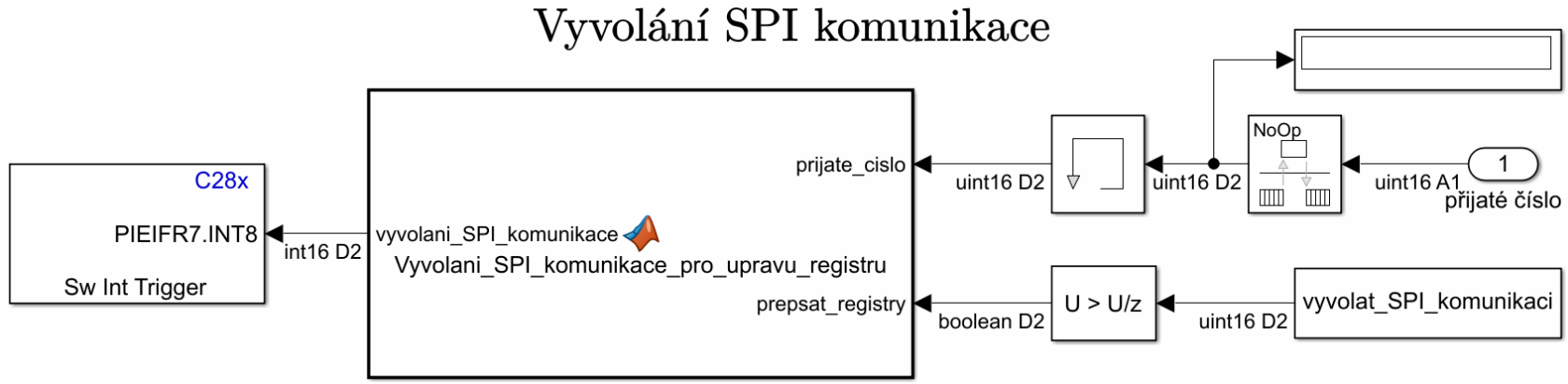

Škálování výstupu
proudové regulace
1
single A2
1
Ta/Tb/Tc
sfix32_En29 (3) A2
A
IQmath
IQNtoF
Y
single (3) A2
+
–
single (3) A2
2250/2
single (3) A2
uint16
uint16 (3) A2
1
CMPA

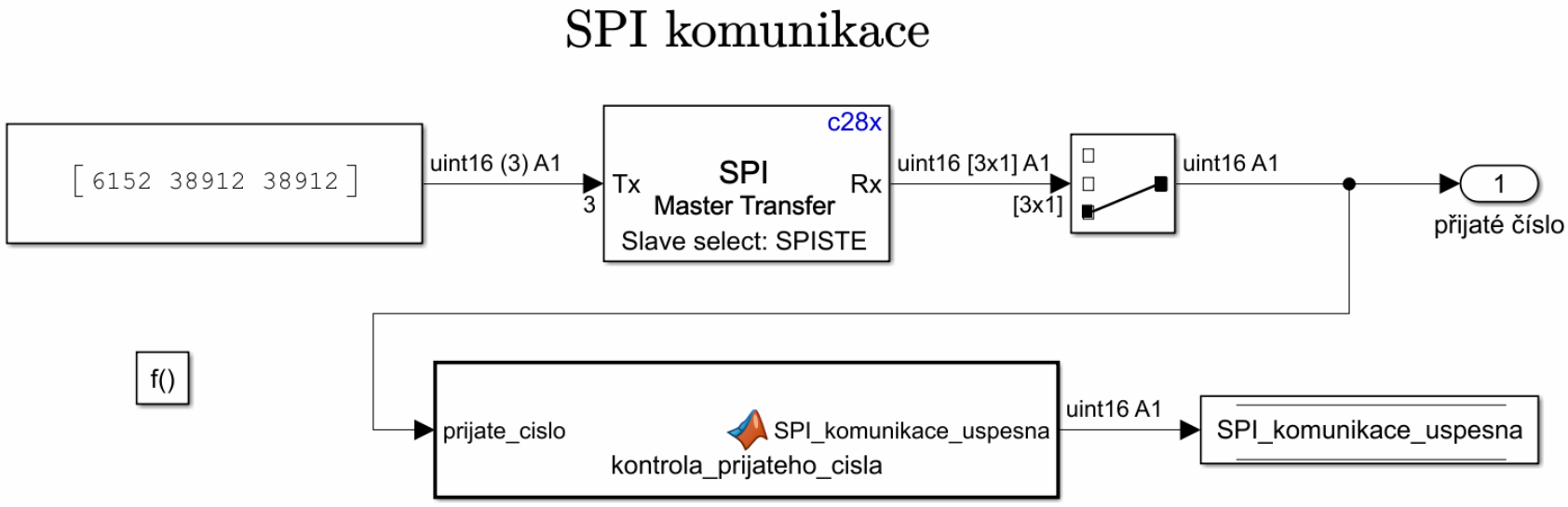

Proudová regulace
0
Požadované id
sfix32_En29 A2
1
id
sfix32_En29 A2
Pozadovane_Iq
sfix32_En29 A2
2
iq
sfix32_En29 A2
3
reset integrátoru
boolean A2
žádaná hodnota
měřená hodnota
reset integrátoru
výstup
proudový PI regulátor - id
sfix32_En29 A2
vd
1
vd
3
požadované iq
žádaná hodnota
měřená hodnota
reset integrátoru
výstup
proudový PI regulátor - iq
sfix32_En29 A2
vq
2
vq

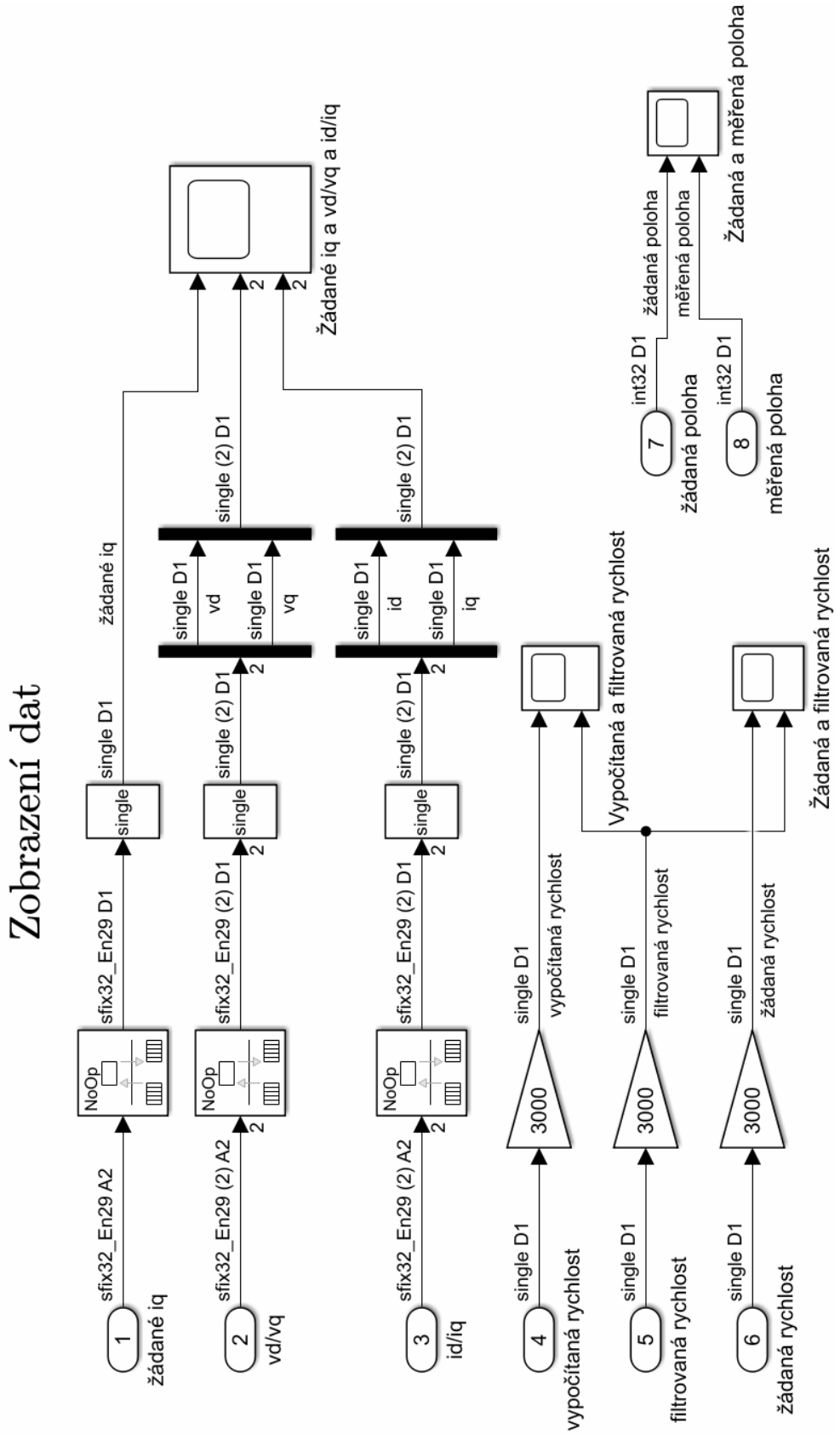

Proudový PI regulátor
PI_Max
H
sfix32_En29 A2
0
max
P složka
IQmath
IQsat
omezení výstupu
výstup
1
žádaná hodnota
2
měřená hodnota
IQNmpy
PI_Ka
Ka_scaled
PI_Kb
Kb_scaled
up
u
lo
y
omezení integrátoru
I složka
min
z-1 u
boolean A2
3
reset integrátoru
PI_Min
L


Rychlostní PI regulátor
PI_Max
H
single D1
0
max
I max
P složka
convert
sfix32_En29 D1
omezení
výstupu
výstup
1
měřená hodnota
2
žádaná hodnota
PI_Kc
PI_Kd
up
u
lo
y
omezení integrátoru
I složka
min
I min
z-1 u
boolean D1
3
reset integrátoru
PI_Min
L


Rychlostní regulace
rozdil_poloh
int32 D1
NoOp
int32 D1
single
single D1
1000*60/8192/3000
Výpočet rychlosti a škálování
single D1
0
single D1
reset_polohy
boolean D1
T
F
single D1
single D1
1
vypočítaná rychlost
single D1
1
žádaná rychlost
0.09516258196404 04
z − 0.90483741803596 0
filtr rychlosti
single D1
měřená hodnota
žádaná hodnota
výstup
reset integrátoru
rychlostní PI regulátor
sfix32_En29 D1
NoOp
sfix32_En29 D1
Pozadovane_lq
boolean D1
NoOp
boolean D1
reset_integratoru
2
filtrovaná rychlost


Vyvolání SPI komunikace
C28x
PIEIFR7.INT8
Sw Int Trigger
int16 D2
vyvolani_SPI_komunikace
Vyvolani_SPI_komunikace_pro_upravu_registru
prijate_cislo
prepsat_registry
uint16 D2
boolean D2
U > U/z
uint16 D2
NoOp
uint16 A1
1
přijaté číslo
vyvolat_SPI_komunikaci


SPI komunikace
[6152 38912 38912]
uint16 (3) A1
3
c28x
Tx
SPI
Master Transfer
Slave select: SPISTE
Rx
uint16 [3x1] A1
[3x1]
uint16 A1
1
přijaté číslo
f()
prijate_cislo
SPI_komunikace_uspesna
kontrola_prijateho_cisla
uint16 A1
SPI_komunikace_uspesna


Zobrazení dat
1
žádané iq
sfix32_En29 A2
NoOp
sfix32_En29 D1
single
single D1
žádané iq
2
vd/vq
sfix32_En29 (2) A2
NoOp
sfix32_En29 (2) D1
single
single (2) D1
single D1
vd
single D1
vq
3
id/iq
sfix32_En29 (2) A2
NoOp
sfix32_En29 (2) D1
single
single (2) D1
single D1
id
single D1
iq
Žádané iq a vd/vq a id/iq
4
vypočítaná rychlost
single D1
3000
vypočítaná rychlost
5
filtrovaná rychlost
single D1
3000
filtrovaná rychlost
Vypočítaná a filtrovaná rychlost
6
žádaná rychlost
single D1
3000
žádaná rychlost
Žádaná a filtrovaná rychlost
7
žádaná poloha
int32 D1
8
měřená poloha
int32 D1
žádaná poloha
měřená poloha
Žádaná a měřená poloha

Příloha B

Literatura

[1] PULLE, Duco W. J., Pete DARNELL a Andre VELTMAN. *Applied Control of Electrical Drives: Real Time Embedded and Sensorless Control using VisSim™ and PLECS™* [online]. Springer International Publishing, ©2015 [cit. 2021-6-19]. ISBN 978-3-319-20043-9. Dostupné z: doi:10.1007/978-3-319-20043-9

[2] DE DONCKER, Rik, Duco W.J. PULLE a André VELTMAN. *Advanced Electrical Drives: Analysis, Modeling, Control* [online]. Springer, ©2011 [cit. 2021-8-4]. ISBN 978-94-007-0181-6. Dostupné z: doi:10.1007/978-94-007-0181-6

[3] WILSON, Dave. TEXAS INSTRUMENTS. *Teaching old motors new tricks with C2000™ – part 1: Introduction to motor control, PI controllers, PID controllers and intro to field oriented control* [online]. [cit. 2021-6-17]. Dostupné z: `https://training.ti.com/teaching-old-motors-new-tricks-part-1-introduction-motor-control-pi-controllers-pid-controllers-and`

[4] WILSON, Dave. TEXAS INSTRUMENTS. *Teaching old motors new tricks with C2000™ – part 2: Field oriented control of permanent magnet synchronous motors* [online]. [cit. 2021-6-19]. Dostupné z: `https://training.ti.com/teaching-old-motors-new-tricks-part-2-field-oriented-control-permanent-magnet-synchronous-motors`

[5] O-ROURKE, Colm J., Mohammad M. QASIM, Matthew R. OVERLIN a James L. KIRTLEY. A Geometric Interpretation of Reference Frames and Transformations: dq0, Clarke, and Park. *IEEE Transactions on Energy Conversion* [online]. Institute of Electrical and Electronics Engineers (IEEE), December 2019 [cit. 2021-6-21]. Dostupné z: `https://hdl.handle.net/1721.1/123557`

[6] ZAMBADA, Jorge a Debraj DEB. MICROCHIP TECHNOLOGY. *AN1078: Sensorless Field Oriented Control of a PMSM*. 2010. ISBN 978-1-60932-099-7. Dostupné z: `http://ww1.microchip.com/downloads/en/Appnotes/01078B.pdf`

[7] MATHWORKS. *C2000 Clarke Transformation: Convert balanced three-phase quantities to balanced two-phase quadrature quantities* [online]. [cit. 2021-6-21]. Dostupné z: `https://www.mathworks.com/help/supportpkg/texasinstrumentsc2000/ref/c2000clarketransformation.html`

[8] MATHWORKS. *C2000 Park Transformation: Convert two-phase stationary system vectors to rotating system vectors* [online]. [cit. 2021-6-23]. Dostupné z: https://www.mathworks.com/help/supportpkg/texasinstrumentsc2000/ref/c2000parktransformation.html

[9] WILSON, Dave. TEXAS INSTRUMENTS. *Teaching old motors new tricks with C2000™ – part 3: Space vector modulation, field Weakening, d-q axis decoupling, and operation of AC induction motors* [online]. [cit. 2021-6-25]. Dostupné z: https://training.ti.com/teaching-old-motors-new-tricks-part-3-space-vector-modulation-field-weakening-d-q-axis-decoupling

[10] TEXAS INSTRUMENTS. *TMS320x2806x Technical Reference Manual* [online]. January 2011–Revised November 2019 [cit. 2021-6-28]. SPRUH18H. Dostupné z: https://www.ti.com/lit/ug/spruh18h/spruh18h.pdf

[11] HOFREITER, Milan. *Základy automatického řízení*. V Praze: České vysoké učení technické, 2012. ISBN 978-80-01-05007-1.

[12] WILSON, Dave. TEXAS INSTRUMENTS. *Teaching Your PI Controller to Behave (Part I)* [online]. July 20, 2015 [cit. 2021-6-28]. Dostupné z: https://e2e.ti.com/blogs_/b/industrial_strength/posts/teaching-your-pi-controller-to-behave-part-i

[13] WILSON, Dave. TEXAS INSTRUMENTS. *Teaching Your PI Controller to Behave (Part II)* [online]. July 20, 2015 [cit. 2021-6-28]. Dostupné z: https://e2e.ti.com/blogs_/b/industrial_strength/posts/teaching-your-pi-controller-to-behave-part-ii

[14] WILSON, Dave. TEXAS INSTRUMENTS. *Teaching Your PI Controller to Behave (Part III)* [online]. March 9, 2013 [cit. 2021-7-5]. Dostupné z: https://e2e.ti.com/blogs_/b/industrial_strength/posts/teaching-you-pi-controller-to-behave-part-iii

[15] WILSON, Dave. TEXAS INSTRUMENTS. *Teaching Your PI Controller to Behave (Part IV)* [online]. March 14, 2013 [cit. 2021-7-5]. Dostupné z: https://e2e.ti.com/blogs_/b/industrial_strength/posts/teaching-your-pi-controller-to-behave-part-iv

[16] WILSON, Dave. TEXAS INSTRUMENTS. *Teaching Your PI Controller to Behave (Part V)* [online]. March 23, 2013 [cit. 2021-8-9]. Dostupné z: https://e2e.ti.com/blogs_/b/industrial_strength/posts/teaching-your-pi-controller-to-behave-part-v

[17] WILSON, Dave. TEXAS INSTRUMENTS. *Teaching Your PI Controller to Behave (Part VII)* [online]. April 13, 2013 [cit. 2021-7-4]. Dostupné z: https://e2e.ti.com/blogs_/b/industrial_strength/posts/teaching-your-pi-controller-to-behave-part-vii

[18] YASKAWA. *Sigma Mini Servo System Product Catalog Supplement* [online]. [cit. 2021-7-6]. Dostupné z: https://www.galco.com/techdoc/yask/sgmm-a1s312_pg.pdf

[19] KUŘINA, Jan. *Řízení synchronního servomotoru pomocí mikroprocesorové platformy Texas Instruments* [online]. Praha, 2019 [cit. 2021-7-6]. Dostupné z: http://hdl.handle.net/10467/84949. Diplomová práce. České vysoké učení technické v Praze, Fakulta strojní, Ústav mechaniky, biomechaniky a mechatroniky. Vedoucí práce Nečas Martin.

[20] TEXAS INSTRUMENTS. *LAUNCHXL-F28069M: C2000 Piccolo MCU F28069M LaunchPad™ development kit* [online]. [cit. 2021-7-6]. Dostupné z: https://www.ti.com/tool/LAUNCHXL-F28069M

[21] TEXAS INSTRUMENTS. *LAUNCHXL-F28069M Overview* [online]. January 2015–Revised March 2019 [cit. 2021-7-6]. SPRUI11B. Dostupné z: https://www.ti.com/lit/ug/sprui11b/sprui11b.pdf

[22] TEXAS INSTRUMENTS. *TMS320F28069M LaunchPad Development Kit Quick Start Guide* [online]. 2014 [cit. 2021-7-6]. SPRUI02. Dostupné z: https://www.ti.com/lit/ml/sprui02/sprui02.pdf

[23] TEXAS INSTRUMENTS. *TMS320F28069M: C2000™ 32-bit MCU with 90 MHz, FPU, VCU, CLA, 256 KB flash, InstaSPIN-MOTION* [online]. [cit. 2021-7-7]. Dostupné z: https://www.ti.com/product/TMS320F28069M

[24] TEXAS INSTRUMENTS. *TMS320F2806x Microcontrollers* [online]. NOVEMBER 2010 – REVISED FEBRUARY 2021 [cit. 2021-7-6]. SPRS698I. Dostupné z: https://www.ti.com/lit/ds/symlink/tms320f28069m.pdf

[25] TEXAS INSTRUMENTS. *BOOSTXL-DRV8301: Motor Drive BoosterPack featuring DRV8301 and NexFET™ MOSFETs* [online]. [cit. 2021-7-7]. Dostupné z: https://www.ti.com/tool/BOOSTXL-DRV8301

[26] TEXAS INSTRUMENTS. *BOOSTXL-DRV8301 Hardware User's Guide* [online]. October 2013 [cit. 2021-7-7]. SLVU974. Dostupné z: https://www.ti.com/lit/ug/slvu974/slvu974.pdf

[27] TEXAS INSTRUMENTS. *Motor Drive BoosterPack Quick Start Guide: BOOSTXL-DRV8301* [online]. 2013 [cit. 2021-7-7]. SLDC006. Dostupné z: https://www.ti.com/lit/eb/sldc006/sldc006.pdf

[28] TEXAS INSTRUMENTS. *DRV8301 Three-Phase Gate Driver With Dual Current Shunt Amplifiers and Buck Regulator* [online]. AUGUST 2011—REVISED JANUARY 2016 [cit. 2021-7-7]. SLOS719F. Dostupné z: https://www.ti.com/lit/ds/symlink/drv8301.pdf

[29] *AX-3005DLS* [online]. [cit. 2021-7-7]. Dostupné z: https://axiomet.eu/cz/cz/product/axiomet/laboratorni-zdroje/ax-3005dls/100165/

[30] *OWON XDS3104AE 100MHz 4 Channel 1GS/s Oscilloscope* [online]. [cit. 2021-7-7]. Dostupné z: https://www.owontechnology.eu/product/1312929/owon-xds3104ae-100mhz-4-channel-1gs-s-oscilloscope

[31] TEXAS INSTRUMENTS. *CCSTUDIO-C2000: Code Composer Studio (CCS) Integrated Development Environment (IDE) for C2000 Microcontrollers* [online]. [cit. 2021-7-7]. Dostupné z: `https://www.ti.com/tool/CCSTUDIO-C2000`

[32] TEXAS INSTRUMENTS. *CONTROLSUITE: controlSUITE™ Software Suite: Software and Development Tools for C2000™ Microcontrollers* [online]. [cit. 2021-7-7]. Dostupné z: `https://www.ti.com/tool/CONTROLSUITE`

[33] TEXAS INSTRUMENTS. *C2000WARE: C2000Ware for C2000 MCUs* [online]. [cit. 2021-7-7]. Dostupné z: `https://www.ti.com/tool/C2000WARE`

[34] MATHWORKS. *What Is MATLAB?* [online]. [cit. 2021-7-7]. Dostupné z: `https://www.mathworks.com/discovery/what-is-matlab.html`

[35] MATHWORKS. *Simulink: Simulation and Model-Based Design* [online]. [cit. 2021-7-8]. Dostupné z: `https://www.mathworks.com/products/simulink.html`

[36] MATHWORKS. *Embedded Coder: Generate C and C++ code optimized for embedded systems* [online]. [cit. 2021-7-8]. Dostupné z: `https://www.mathworks.com/products/embedded-coder.html`

[37] MATHWORKS. *Embedded Coder Support Package for Texas Instruments C2000 Processors* [online]. [cit. 2021-7-8]. Dostupné z: `https://www.mathworks.com/matlabcentral/fileexchange/43096-embedded-coder-support-package-for-texas-instruments-c2000-processors`

[38] MATHWORKS. *C280x/C2802x/C2803x/C2805x/C2806x/C2833x/C2834x/F28M3x/F2807x/F2837xD/F2837xS/F2838x/F28004x/F28002x ePWM: Generate enhanced Pulse Width Modulated (ePWM) waveforms* [online]. [cit. 2021-7-12]. Dostupné z: https://www.mathworks.com/help/supportpkg/texasinstrumentsc2000/ref/c280xc2802xc2803xc2805xc2806xc2833xc2834xf28m3xf2807xf2837xdf2837xsf2838xf28004xf28002xepwm.html

[39] TEXAS INSTRUMENTS. *User's Guide: InstaSPIN-FOC™ and InstaSPIN-MOTION™* [online]. January 2013–Revised June 2019 [cit. 2021-7-14]. SPRUHJ1H. Dostupné z: `https://www.ti.com/lit/ug/spruhj1h/spruhj1h.pdf`

[40] TEXAS INSTRUMENTS. *BOOSTXL-DRV8301 Hardware Files: BOOSTXL-DRV8301_SCH* [online]. [cit. 2020-11-8]. SLVC539. Dostupné z: `https://www.ti.com/lit/df/slvc539/slvc539.zip`

[41] MATHWORKS. *C2802x/C2803x/C2805x/C2806x/F28M3x/F2807x/F2837xD/F2837xS/F2838x/F28004x/F28002x ADC: Configure ADC to sample analog pins and output digital data* [online]. [cit. 2021-7-14]. Dostupné z: https://www.mathworks.com/help/supportpkg/texasinstrumentsc2000/ref/c2802xc2803xc2805xc2806xf28m3xf2807xf2837xdf2837xsf2838xf28004xf28002xadc.html

[42] MATHWORKS. *C28x eQEP: Quadrature encoder pulse block used to derive position, direction, and speed* [online]. [cit. 2021-7-18]. Dostupné

z: https://www.mathworks.com/help/supportpkg/texasinstrumentsc2000/ref/c28xeqep.html

[43] MATHWORKS. *Data Types Supported by Simulink* [online]. [cit. 2021-7-18]. Dostupné z: https://www.mathworks.com/help/simulink/ug/data-types-supported-by-simulink.html

[44] MATHWORKS. *Data Type Support* [online]. [cit. 2021-7-18]. Dostupné z: https://www.mathworks.com/help/supportpkg/texasinstrumentsc2000/ug/data-type-support.html

[45] MATHWORKS. *Fixed-Point Data in MATLAB and Simulink* [online]. [cit. 2021-7-18]. Dostupné z: https://www.mathworks.com/help/simulink/ug/fixed-point-data-in-matlab-and-simulink.html

[46] MATHWORKS. *Fixdt: Create Simulink.NumericType object describing a fixed-point or floating-point data type* [online]. [cit. 2021-7-18]. Dostupné z: https://www.mathworks.com/help/simulink/slref/simulink.numerictype.fixdt.html

[47] MATHWORKS. *Data Types and Scaling in Digital Hardware* [online]. [cit. 2021-7-18]. Dostupné z: https://www.mathworks.com/help/fixedpoint/ug/data-types-and-scaling-in-digital-hardware.html

[48] TEXAS INSTRUMENTS. *TMS320F2806x Microcontrollers* [online]. NOVEMBER 2010 – REVISED FEBRUARY 2021 [cit. 2021-7-18]. SPRS698I. Dostupné z: https://www.ti.com/lit/ds/sprs698i/sprs698i.pdf

[49] MATHWORKS. *Using the IQmath Library* [online]. [cit. 2021-7-21]. Dostupné z: https://www.mathworks.com/help/supportpkg/texasinstrumentsc2000/ug/using-the-iqmath-library.html

[50] MATHWORKS. *Specify Sample Time* [online]. [cit. 2021-7-26]. Dostupné z: https://www.mathworks.com/help/simulink/ug/how-to-specify-the-sample-time.html

[51] MATHWORKS. *Types of Sample Time* [online]. [cit. 2021-7-26]. Dostupné z: https://www.mathworks.com/help/simulink/ug/types-of-sample-time.html

[52] MATHWORKS. *C28x Hardware Interrupt: Interrupt Service Routine to handle hardware interrupt on C28x processors* [online]. [cit. 2021-8-8]. Dostupné z: https://www.mathworks.com/help/supportpkg/texasinstrumentsc2000/ref/c28xhardwareinterrupt.html

[53] MATHWORKS. *C2000 Trig Fcn IQN: Sine, cosine, or arc tangent of IQ number* [online]. [cit. 2021-7-20]. Dostupné z: https://www.mathworks.com/help/supportpkg/texasinstrumentsc2000/ref/c2000trigfcniqn.html

[54] WILSON, Dave. TEXAS INSTRUMENTS. *Teaching Your PI Controller to Behave (Part VI)* [online]. April 5, 2013 [cit. 2021-8-9]. Dostupné z: https://e2e.ti.com/blogs_/b/industrial_strength/posts/teaching-your-pi-controller-to-behave-part-vi

[55] WILSON, Dave. TEXAS INSTRUMENTS. *Teaching Your PI Controller to Behave (Part X)* [online]. May 15, 2013 [cit. 2021-8-9]. Dostupné z: https://e2e.ti.com/blogs_/b/industrial_strength/posts/teaching-your-pi-controller-to-behave-part-x

[56] WILSON, Dave. TEXAS INSTRUMENTS. *Teaching Your PI Controller to Behave (Part IX)* [online]. April 30, 2013 [cit. 2021-8-9]. Dostupné z: https://e2e.ti.com/blogs_/b/industrial_strength/posts/teaching-your-pi-controller-to-behave-part-ix

[57] TEXAS INSTRUMENTS. *[FAQ] Sticky: Enable XDS100 USB Serial Port on LAUNCHXL-F28069M* [online]. [cit. 2021-8-9]. Dostupné z: https://e2e.ti.com/support/microcontrollers/c2000-microcontrollers-group/c2000/f/c2000-microcontrollers-forum/670519/faq-sticky-enable-xds100-usb-serial-port-on-launchxl-f28069m

[58] MATHWORKS. *Parameter Tuning and Signal Logging over Serial Communication* [online]. [cit. 2021-8-9]. Dostupné z: https://www.mathworks.com/help/supportpkg/texasinstrumentsc2000/ug/external-mode.html

[59] MATHWORKS. *External Mode Simulation by Using XCP Communication* [online]. [cit. 2021-8-9]. Dostupné z: https://www.mathworks.com/help/rtw/ug/external-mode-simulation-with-xcp-communication.html

[60] MATHWORKS. *Default parameter behavior* [online]. [cit. 2021-8-9]. Dostupné z: https://www.mathworks.com/help/rtw/ref/default-parameter-behavior.html

[61] MATHWORKS. *Create Tunable Calibration Parameter in the Generated Code* [online]. [cit. 2021-8-9]. Dostupné z: https://www.mathworks.com/help/rtw/ug/use-parameter-objects-for-code-generation.html