ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ

DIPLOMOVÁ PRÁCA

Podpora hardware periferií při generování kódu ze Simulinku

Praha, 2008 Tomáš Bysterský

České vysoké učení technické v Praze - Fakulta elektrotechnická

Katedra řídicí techniky *Školní rok:* 2006/2007

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Tomáš Bysterský

Obor: Technická kybernetika

Název tématu: Podpora hardware periférií při generování kódu ze Simulinku

Zásady pro vypracování:

1. Seznamte se s technologií generování kódu ze Simulinku a s nástrojem Processor Expert a běžnými periferiemi mikrokontrolerů.
2. Implementuje simulační bloky reprezentující vybranou skupinu periférií mikrokontroleru.
3. Pro vytvořené simulační bloky vytvořte podporu pro generování kódu obsluhující tyto periférie.
4. Na vhodném příkladě demonstrujte použití bloků.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí diplomové práce: Ing. Libor Waszniowski, Ph.D.

Termín zadání diplomové práce: zimní semestr 2006/2007

Termín odevzdání diplomové práce: leden 2008

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

v zast.
prof. Ing. Zbyněk Škvor, CSc.
děkan

V Praze dne 21.02.2007

Poďakovanie

Rád by som poďakoval vedúcemu diplomovej práce Ing. Liborovi Waszniowskému, Ph.D., za poskytnuté odborné rady a podporu pri vypracovaní diplomovej práce.

Prehlásenie:

Prehlasujem, že som svoju diplomovú prácu vypracoval samostatne a použil som iba podklady (literatúru, projekty, SW atd.) uvedené v priloženom zozname.

V Prahe dňa 18. 1. 2008 ……

……

podpis

Abstrakt:

Diplomová práca sa zaoberá problematikou simulácie hardware periférií a automatickým generovaním kódu, využívaných pri návrhu embedded zariadení metódou rapid prototyping. V práci sú spomenuté výhody a princípy automatického generovania kódu z prostredií Matlab/Simulink a Processor Expert, pomocou knižnice Peslib. Knižnica Peslib bola rozšírená o blok PWMMC.

Použitie a funkčnosť blokov knižnice Peslib je ukázané na demonštračnej úlohe riadenia BLDC motora. Úloha zahŕňa modelovanie, simulovanie a implementáciu riadiacich algoritmov s následným automatickým generovaním C kódu regulátora cieľovej aplikácie.

Abstract:

Diploma thesis describes problems of hardware peripherals simulation and automatic code generation, applied in the development of embedded devices using a method of rapid prototyping. The thesis discuses the advantages and principles of automatic code generation in Matlab/Simulink and Processor Expert while using Peslib block-set. The Peslib blockset has been extended by block PWMMC.

Usage and operation of the Peslib block-set is demonstrated on an example of the BLDC motor control process. This task includes modeling, simulation and implementation of control algorithms followed by automatic C code generation for the target application controller.

Obsah

1 ÚVOD ... 6

2 POUŽITÉ SW NÁSTROJE ... 8

2.1 MATLAB-SIMULINK ... 8

2.1.1 S-funkcie ... 9

2.1.1.1 Priebeh volania metód S-funkcii počas simulácie ... 9

2.1.2 RTW, RTW EC ... 11

2.1.2.1 Automatické generovanie C kódu zo Simulinku ... 11

2.1.2.2 TLC súbor s výsledným automaticky generovaným C kódom ... 12

2.1.2.3 Nastavenie RTW ... 13

2.1.3 StateFlow ... 14

2.1.4 Fixed-Point settings ... 15

2.2 PROCESSOR EXPERT ... 16

2.3 PESLIB A PEERT ... 18

2.3.1 Podpora HW periférií knižnicou Peslib ... 18

2.3.2 Tvorba projektu s využitím Peslib knižnice, generovanie kódu aplikácie. ... 19

2.4 VIZUALIZAČNÉ PROSTREDIE FREEMASTER ... 21

3 ROZŠÍRENIE PODPORY HW PERIFÉRIÍ PESLIB KNIŽNICE ... 22

3.1 IMPLEMENTÁCIA S-FUNKCIE PWMMC ... 22

3.2 IMPLEMENTÁCIA TLC SÚBORU PWMMC ... 23

4 POPIS DEMONŠTRAČNEJ ÚLOHY ... 25

4.1 POROVNANIE BLDC A PMSM MOTOROV ... 25

4.1.1 Matematický popis BLDC motora ... 27

4.1.2 Simulačná schéma BLDC motora ... 29

4.2 DIGITÁLNE RIADENIE BLDC MOTORA ... 30

4.2.1 Meranie polohy rotora BLDC ... 32

4.2.1.1 Hallové senzory ... 32

4.2.1.2 IRC senzor s Quadrature dekodérom ... 33

4.2.2 Komutácia BLDC motora ... 34

4.2.2.1 Komutácia BLDC s použitím Hall Senzorov ... 34

4.2.2.2 Komutácia BLDC s použitím IRC a QD ... 35

4.2.3 Meranie frekvencie otáčok BLDC rotora ... 37

4.2.3.1 Meranie frekvencie otáčok rotora s použitím IRC a QD ... 37

4.2.3.2 Meranie frekvencie otáčok rotora s použitím Hall senzorov ... 39

4.2.4 Regulátor frekvencie otáčania rotora BLDC ... 39

4.3 POUŽITÝ HW ... 40

4.3.1 Vývojová doska MC56F8367EVM ... 41

4.3.2 EVM Motor Board ... 43

4.4 IMPLEMENTÁCIA RIADIACEHO ALGORITMU BLDC MOTORA 44
4.4.1 Riadenie BLDC so spätnou väzbou od IRC a QD 44
4.4.2 Riadenie BLDC so spätnou väzbou od Hall senzorov 47
5 VYHODNOTENIE SIMULÁCIE A REÁLNEHO RIADENIA BLDC 50
5.1 SIMULÁCIA RIADENIA BLDC SO SPÄTNOU VÄZBOU OD IRC A QD 50
5.2 REÁLNA REGULÁCIA BLDC SO SPÄTNOU VÄZBOU OD IRC A QD 51
5.3 SIMULÁCIA RIADENIA BLDC SO SPÄTNOU VÄZBOU OD HALL SENZOROV 53
5.4 REÁLNA REGULÁCIA BLDC SO SPÄTNOU VÄZBOU OD HALL SENZOROV 56
6 ZÁVER 59
7 SLOVNÍK POUŽITÝCH TERMÍNOV A SKRATIEK 61
8 POUŽITÁ LITERATÚRA A ZDROJE: 62
PRÍLOHA A. MATEMATICKÝ POPIS A SIMULAČNÁ SCHÉMA PMSM 64
PRÍLOHA B. POPIS S-FUNKCIE PWMMC 66
PRÍLOHA C. NASTAVENIE KONEKTOROV PROCESOROVEJ DOSKY MC56F8367EVM .. 67
PRÍLOHA D. ŠTRUKTÚRA REGULÁTORA 70
PRÍLOHA E. VIZUALIZÁCIA PROCESU RIADENIA BLDC V PROGRAME FREEMASTER71
PRÍLOHA F. FOTODOKUMENTÁCIA 73

Zoznam obrázkov

OBR. 1 PRIEBEH VOLANIA POVINNÝCH METÓD S-FUNKCIÍ 10
OBR. 2 AUTOMATICKÉ GENEROVANIE C KÓDU ZO SIMULINKU 12
OBR. 3 TLC SÚBOR S VYGENEROVANÝM C KÓDOM ZO SIMULINKU 13
OBR. 4 DIALÓGOVÉ OKNO CONFIGURATION PARAMETERS 14
OBR. 5 STATEFLOW EDITOR 14
OBR. 6 OBECNÉ FIXED-POINT ČÍSLO [19] 16
OBR. 7 PROCESSOR EXPERT PROJEKT S POUŽITÝMI EMBEDDE BEANS 17
OBR. 8 PESLIB BLOCK SET 19
OBR. 9 POSTUP TVORENIA APLIKÁCIE S VYUŽITÍM PEERT V SIMULINKU A PE 21
OBR. 10 UKÁŽKA FUNKČNOSTI PWMMC 23
OBR. 11 REZ BLDC MOTORA [7] 25
OBR. 12 ROZLOŽENIE BEMF BLDC A PMSM MOTORA [16] 26
OBR. 13 SIMULAČNÝ MODEL BLDC MOTORA 30
OBR. 14 APLIKOVANÉ SVORKOVÉ NAPÄTIA PRI TVORBE ROTUJÚCEHO EM POĽA STATORA 31
OBR. 15 IDENTIFIKÁCIA POLOHY ROTORA HALL SENZORMI 33
OBR. 16 PRINCÍP FUNKCIE IRC SENZORA S QD DEKODÉROM [11] 34
OBR. 17 STAV PRIPOJENÝCH SVORKOVÝCH NAPÄTÍ PRED A PO KOMUTÁCII 35
OBR. 18 HRANICE KOMUTÁCIE S VYUŽITÍM QD 36
OBR. 19 NASTAVOVANIE COMPARE REGISTROV QD PRI DETEKCIÍ KOMUTÁCIE 37
OBR. 20 PI REGULÁTOR FREKVENCIE OTÁČANIE ROTORA 40
OBR. 21 VÝVOJOVÁ DOSKA MC56F8367EVM S VÝKONOVOU ČASŤOU A BLDC MOTOROM 41
OBR. 22 BLOKOVÁ SCHÉMA 56F8367 42
OBR. 23 TROJFÁZOVÝ H-MOSTÍK [7] 43
OBR. 24 KOMPLEMENTÁRNE SPÍNANIE TRANZISTOROV H-MOSTÍKA S VLOŽENÝM DEAD TIME 44
OBR. 25 SIMULAČNÁ SCHÉMA RIADENIA BLDC SO SPÄTNOU VÄZBOU OD IRC A QD 45
OBR. 26 SIMULAČNÁ SCHÉMA SUBSYSTÉMU REGULÁTORA SO SPÄTNOU VÄZBOU OD IRC A QD 46
OBR. 27 SIMULAČNÁ SCHÉMA RIADIACEHO ALGORITMU BLDC SO SPÄTNOU VÄZBOU OD IRC A QD 47
OBR. 28 SIMULAČNÁ SCHÉMA RIADENIA BLDC SO SPÄTNOU VÄZBOU OD HALL SENZOROV 48
OBR. 29 SIMULAČNÁ SCHÉMA RIADENIA BLDC SO SPÄTNOU VÄZBOU OD HALL SENZORA 49
OBR. 30 SIMULAČNÁ SCHÉMA RIADIACEHO ALGORITMU BLDC SO SPÄTNOU VÄZBOU OD HALL SENZORA 49

OBR. 31 SIMULOVANÝ VÝSTUP REGULOVANIA FREKVENCIE OTÁČANIA ROTORA, IRC SENZOR 50
OBR. 32 SIMULOVANÝ KRÚTIACI MOMENT ZÁŤAŽE A ELE. KRÚTIACI MOMENT MOTORA, IRC SENZOR 51
OBR. 33 SIMULOVANÝ AKČNÝ ZÁSAH PI REGULÁTORA, IRC SENZOR 51
OBR. 34 VÝSLEDKY MERANIA FREKVENCIE BLDC, IRC SENZOR 52
OBR. 35 VÝSLEDKY MERANIA AKČNÉHO ZÁSAHU PI REGULÁTORA, IRC SENZOR 52
OBR. 36 SIMULOVANÝ VÝSTUP REGULOVANIA FREKVENCIE OTÁČANIA ROTORA, HALL SENZOR 53
OBR. 37 SIMULOVANÝ KRÚTIACI MOMENT ZÁŤAŽE A ELE. KRÚTIACI MOMENT MOTORA, HALL SENZOR 54
OBR. 38 SIMULOVANÝ AKČNÝ ZÁSAH PI REGULÁTORA, HALL SENZOR 54
OBR. 39 SIMULOVANÝ VÝSTUP REGULOVANIA FREKVENCIE OTÁČANIA ROTORA, HALL SENZOR, F_{REQ}=1HZ 55
OBR. 40 SIMULOVANÝ KRÚTIACI MOMENT ZÁŤAŽE A ELE. KRÚTIACI MOMENT MOTORA, HALL SENZOR, F_{REQ}=1HZ 55
OBR. 41 SIMULOVANÝ AKČNÝ ZÁSAH PI REGULÁTORA, HALL SENZOR, F_{REQ}=1HZ 55
OBR. 42 VÝSLEDKY MERANIA FREKVENCIE BLDC, HALL SENZOR 56
OBR. 43 VÝSLEDKY MERANIA AKČNÉHO ZÁSAHU PI REGULÁTORA, HALL SENZOR 56
OBR. 44VÝSLEDKY MERANIA FREKVENCIE BLDC, HALL SENZOR, F_{REQ}=1HZ 57
OBR. 45 VÝSLEDKY MERANIA AKČNÉHO ZÁSAHU PI REGULÁTORA, HALL SENZOR, F_{REQ}=1HZ 57
OBR. 46 SIMULAČNÁ SCHÉMA PMSM MOTORA 65
OBR. 47 ROZLOŽENIE KONEKTOROV VÝVOJOVEJ DOSKY MC56F8367EVM 67
OBR. 48 ROZLOŽENIE KONEKTOROV LEGACY MOTOR INTERFACE DAUGHTER CARD (LMIDC) 68
OBR. 49 VÝKONOVÁ ČASŤ EVM MOTOR BOARD (EVM) 69
OBR. 50 ŠTRUKTÚRA CTRL_ALG SO SPÄTNOU VÄZBOU OD IRC A QD 70
OBR. 51 ŠTRUKTÚRA CTRL_ALG SO SPÄTNOU VÄZBOU OD HALL SENZOROV 70
OBR. 52 OVLÁDACÍ PANEL VIZUALIZÁCIE RIADENIA BLDC MOTORA 71
OBR. 53 MERANIE FREKVENCIE OTÁČOK ROTORA 71
OBR. 54 MERANIE AKČNÝCH ZÁSAHOV PI REGULÁTORA 72
OBR. 55 ZDRUŽENÉ MERANIE FREKVENCIE OTÁČOK ROTORA A AKČNÝCH ZÁSAHOV PI REGULÁTORA 72
OBR. 56 BLDC MOTOR, RIADIACI SYSTÉM, VIZUALIZÁCIA PROCESU 73

Zoznam tabuliek

TAB. 1: PARAMETRE MOTORA MCG IB-23811 30
TAB. 2: KOMUTAČNÉ SEKVENCIE BLDC MOTORA..... 35
TAB. 3 NASTAVENIE KONEKTOROV VÝVOJOVEJ DOSKY MC56F8367EVM..... 67
TAB. 4 NASTAVENIE KONEKTOROV LEGACY MOTOR INTERFACE DAUGHTER CARD 68

1 Úvod

V súčasnej dobe, pri vzrastajúcom dopyte a využívaní mikroprocesorovej riadiacej techniky, sa zvyšujú nároky na rýchlosť vývoja a implementáciu nových hardwarových a softwarových riešení.

S využitím tradičných spôsobov návrhu embedded zariadení v riadiacej technike, pri ktorých obvykle spolupracuje viacej špecializovaných tímov, zameraných na návrh riadiacich algoritmov, softwarových riešení riadiacich algoritmov, hardwarových riešení cieľovej platformy a implementačného tímu, je časovo a finančne neefektívny. Tradičný návrhu embedded zariadení počíta s postupom, kde tím navrhujúci riadiace algoritmy vytvorí riešenie, ktoré softwarový špecialisti manuálne naprogramujú do simulačného prostredia, z ktorého vyplynú hardwarové požiadavky cieľovej platformy. Po vytvorení hardwaru, sa ručne implementuje softwarové riešenie riadenia a cieľový produkt sa pošle na validačný proces, ktorého úspešnosť určí, či sa celý proces bude opakovať. Pri neúspechu testovania sa opakovaním procesu návrhu zvyšuje čas a cena potrebná k uvedeniu nového produktu na trh. Nevýhody takéhoto spôsobu vývoja embedded zariadení je v skutočnosti, že kvalita produktu je ovplyvnená ľudským faktorom pri písaní kódu aplikácie a absenciou priameho spojenia od návrhu k implementácií.

Riešením nevýhod tradičného návrhu embedded zariadení, je technika rýchleho iterovania rapid prototypingu, ktorá spojuje proces algoritmického, softwarového a hardwarového návrhu, do jedného kroku s možnosťou odhalenia potencionálnych problémov riešenia, už vo fáze simulácie. Jedným z takýchto nástrojov rapid prototipingu je spojenie prostredí Matlab/Simulink a Processor Expert (PE), s podporou automatického generovania produkčného kódu nadstavbou Real Time Workshop (RTW) pre Processor Expert Embedded Real-Time Target (PEERT), ktorý je časťou knižnice Peslib vyvíjanej v rámci projektu UTIA, UNIS a Katedry řídicí techniky (ČVUT, FEL).

Základom Rapid Prototypingu v spomenutom prostredí, je vo fáze simulácie a modelovania používaný block-setu knižnice Peslib, ktorým simulujeme rozhranie embedded systému s riadeným procesom v simulinkovej schéme. Algoritmická časť riešenia procesu embedded zariadením sa zabezpečuje tvorbou regulátora klasickými metódami riadenia v prostredí Simulinku. Hardwarová časť implementácie je určená

použitím block-setu knižnice Peslib a softwarová časť implementácie je zabezpečená automatickým generovaním kódu z časti regulátora a blokov knižnice Peslib v RTW a PE, s možnosťou aplikovania vygenerovaného kódu k programovaniu embedded zariadenia. Automaticky generovaný kód je pritom získaný v kratšom čase a nie je zaťažený ľudským faktorom chybovosti pri programovaní.

Cieľom tejto práce je oboznámiť sa s technológiou simulácie hardware periférií mikroprocesorov, s podporou automatického generovania kódu zo Simulinku a Processor Expertu. S využitím týchto znalostí chceme rozšíriť block-set knižnice Peslib o blok PWMMC, potrebný pre riadení trojfázových motorov, s podporou automatického generovania kódu. Najväčšia časť práce ukazuje spôsob použitia knižnice Peslib pri vývoji software regulátora otáčok bezkartáčového jednosmerného (BLDC) motora.

2 Použité SW nástroje

V rámci zadanej práce bolo použité integrované prostredia Matlab s nadstavbou Simulink. Slúžili k simulácii chovania vstupných a výstupných HW periférii mikrokontrolérov a k simuláciám riadiacich algoritmov aplikovaných na modeli BLDC motora. Automatické generovanie kódu z časti regulátora zabezpečovali Real Time Workshop (RTW) a Real Time Workshop Embedded Coder (RTW EC), ktoré sú ich súčasťou. Pri implementácií riadiacich algoritmov bol využitý StateFlow editor k vytváraniu automatov riadiacej logiky.

Samotné programovanie výsledného produktu bolo zabezpečené vývojovým prostredím CodeWarrior (CW), ktorého súčasťou je podporné prostredie Processor Expert (PE). Processor Expert je určený k rýchlej konfigurácií a programovaniu jednotlivých typov mikrokontrolérov bez podrobnej znalosti ich HW architektúry. K výslednému produktu bola vytvorená vizualizácia v prostredí FreeMaster.

Komunikáciu medzi prostredím Simulinku a Processor Expertu zabezpečuje knižnica Peslib, ktorá je vyvíjaná v rámci projektu UTIA. Hlavnou úlohou Peslib je synchronizácia premenných parametrov HW periférii nastavovaných v Processor Experte s parametrami ktoré ovplyvňujú simuláciu HW periférií testovaného projektu. Generovanie kódu zaisťuje Processor Expert Embedded Real Time Target (PEERT) ktorý je vyvíjaný na Katedře řídicí techniky FEL ČVUT.

2.1 Matlab-Simulink

Matlab je integrované prostredie od spoločnosti Mathworks [19] určené k vykonávaniu vedecko-technických výpočtov, návrhu algoritmov, modelovaniu a simulovaniu dynamických systémov s možnosťou prezentácie výstupných dát.

Simulink je nadstavba Matlabu, ktorá je určená k modelovaniu a simulovaniu časovo premenných dynamických systémov. Simulink obsahuje knižnice tematicky rozdelených grafických blokov (block set), z ktorých je možné jednoducho vytvárať modely dynamických sústav, vo forme blokových schém a rovníc. S pomocou Simulinku a jeho grafického editoru je možné vytvárať lineárne, nelineárne, v čase diskrétne a spojité systémy.

2.1.1 S-funkcie

Pri nedostatočnom pokrytí simulovaného problému štandardnou blokovou sadou Simulinku, je možné vytvoriť si vlastný block set s využitím S-funkcií. K programovaniu S-funkcií je možné použiť jazyky ako Matlab, C, C++, Ada a Fortran. Štruktúra zdrojového kódu S-funkcie je pevne daná. Obsahuje skupinu definovaných metód, ktoré sú volané Simulinkom v jednotlivých fázach vykonávania simulácie. Časový priebeh volania povinných metód S-funkcií naprogramovaných v jazyku C, je vysvetlený v odstavci 2.1.1.1.

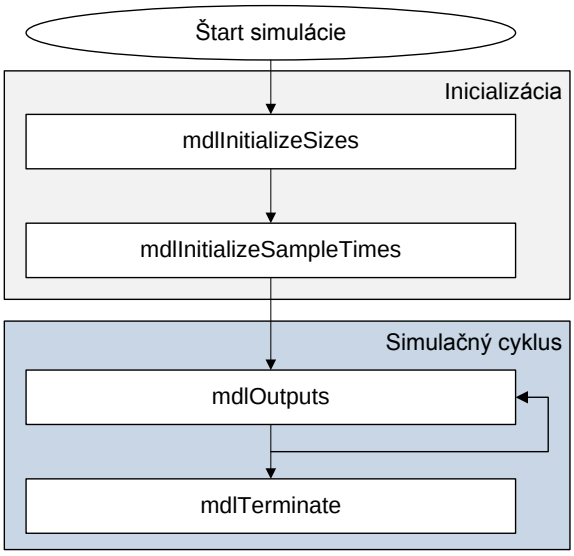
V rámci diplomovej práce bol použitý k programovaniu S-funkcií jazyk C. Samotné zdrojové kódy S-funkcií, je nutné pred použitím v Simulinku preložiť do formy dynamickej knižnice s príponou **.mexw32*. Preklad zdrojových kódov dosiahneme zadaním príkazu *mex sfname.c* do príkazového okna Matlabu. Podrobné informácie k programovaniu S-funkcií sú uvedené v dokumente [1]. Hlavné výhody využívania S-funkcií, pri vytváraní vlastných block setov v jazyku C sú:

- možnosť navrhnutia vlastných algoritmov
- zvýšenie výpočtového výkonu programovaných funkcií
- možnosť ovplyvnenia automaticky generovaného kódu zo Simulinku, pomocou TLC súborov (Target Language Compiler).

2.1.1.1 Priebeh volania metód S-funkcii počas simulácie

Časový priebeh volania povinných metód S-funkcie je znázornený na Obr. 1. Priebeh simulácie sa dá rozdeliť na inicializačnú časť, ktorá sa vykonáva len raz pri spustení simulácie a samotného simulačného cyklu, volaného opakovane počas celej simulácie.

Pri spustení automatického generovania kódu zo Simulinku prostredníctvom RTW, je priebeh volania povinných metód S-funkcií odlišný. Pri generovaní kódu je po vykonaní inicializáčnej časti bloku, volaná Simulinkom metóda mdlRTW. Proces volania metód Simulinkom je ukončený metódou mdlTerminate.



Obr. 1 Priebeh volania povinných metód S-funkcií

Inicializácia:

- *mdlInitializeSizes*: metóda slúži k nastaveniu počtu vstupných a výstupných portov, ich šírkam, k nastaveniu vstupných a výstupných dátových typov a k alokácií pamäte pomocných premenných používaných blokom počas simulačného cyklu.
- *mdlInitializeSampleTimes*: metóda slúži k nastaveniu vzorkovacích časov vstupných a výstupných portov programovaného bloku.

Simulačný cyklus:

- *mdlOutputs*: metóda sa používa k výpočtu výstupov simulovaného bloku s využitím užívateľom naprogramovaného algoritmu.
- *mdlTerminate*: metóda je volaná pri skončení simulácie. Slúži k uvoľneniu pamäte užívateľom zadefinovaných premenných využívaných počas simulácie.

Generovanie kódu:

- mdlRTW: metóda je volaná Simulinkom pri spustení generovania kódu prostredníctvom RTW. Volaná metóda sa používa k vloženiu informácií o danom bloku do súboru *model.rtw*, ktorý je spracovávaný Target Language Compilerom (TLC) pri generovaní kódu zo Simulinku.

2.1.2 RTW, RTW EC

Real Time Workshop (RTW) [2] je rozšírením prostredia Matlab-Simulink, ktoré automaticky generuje kód zo simulinkových projektov pre zadaný Target. Target reprezentuje cieľovú HW a SW platformu, pre ktorú je zo simulinkového projektu generovaný výsledný kód.

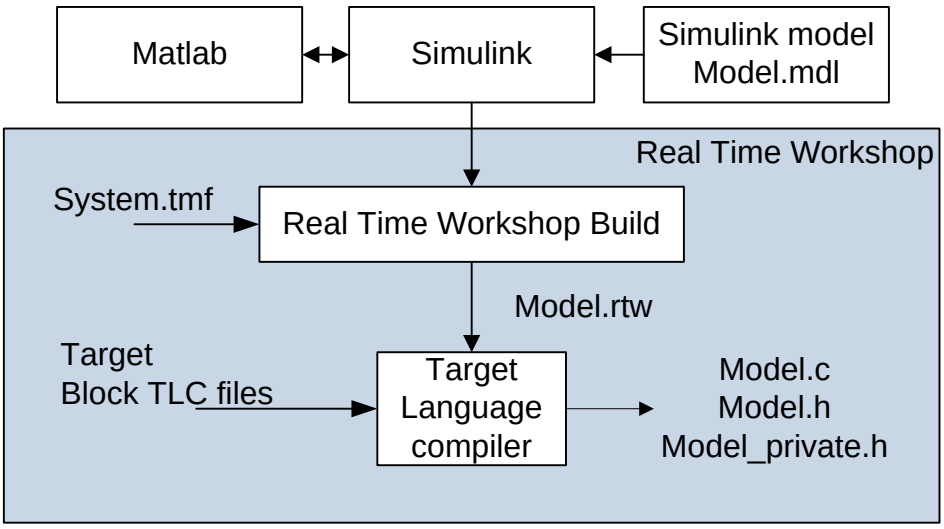
Real Time Workshop Embedded Coder (RTW EC) [3] je nadstavba RTW, ktorá je špecializovaná k vývoju embedded zariadení. RTW EC umožňuje generovať produkčný ANSI-C, poprípade ISO-C kód, ktorý je vysoko optimalizovaný na rýchlosť, pamäťovú náročnosť a čitateľnosť kódu. RTW EC podporuje generovanie kódu pre cieľové mikroprocesory s fixed-point a floating-point aritmetikou.

2.1.2.1 Automatické generovanie C kódu zo Simulinku

Priebeh automatického generovania C kódu zo Simulinku a RTW je znázornený na Obr. 2. Po spustení generovania kódu zo Simulinku, RTW spracuje popis modelu simulinkovej schémy *Model.mdl* do súboru *Model.rtw*. Súbor *Model.rtw* obsahuje hierarchickú štruktúru blokov obsiahnutých v simulinkovej schéme, s ich vzájomnými prepojeniami potrebných k ďalšiemu spracovaniu. Target Language Compiler (TLC) následne spracuje súbor *Model.rtw*, z ktorého vytvorí C kód odpovedajúci simulinkovému modelu.

Ovplyvniť spôsob generovania výsledného C kódu je možné dvoma spôsobmi:

- Vytvorením vlastného *system target súboru* ktorý ovplyvňuje celkové parametre generovaného kódu
- Vytvorením vlastných TLC súborov priradených blokom používaných pri generovaní kódu. Ukážka TLC súboru s vygenerovaným C kódom je v odstavci 2.1.2.2.



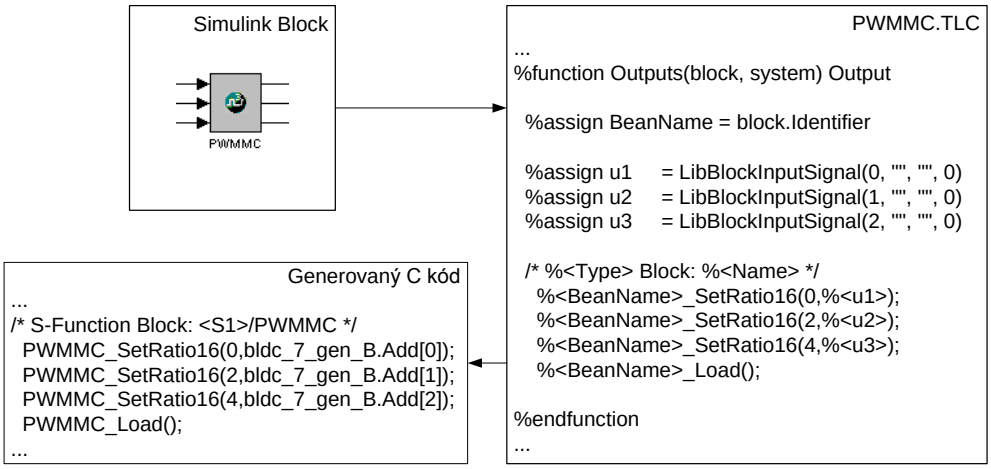
Obr. 2 Automatické generovanie C kódu zo Simulinku

Spustiť automatické generovanie kódu zo Simulinku je možné viacerými spôsobmi:

- stlačením tlačidla „Generate code“ v záložke „Real-Time Workshop“ v simulinkovom dialógovom okne „Configuration Parameters”
- stlačením klávesovej skratky „Ctrl+B” v simulinkovom modele
- zadaním príkazu „rtwbuild(‘Model’)“ v príkazovom okne Matlabu

2.1.2.2 TLC súbor s výsledným automaticky generovaným C kódom

Prostredníctvom TLC súborov je možné presne zadefinovať, akým spôsobom prebehne generovanie výsledného C kódu pre daný blok v simulinkovej schéme. TLC súbor definuje, aké časti kódu a v akom tvare sa budú vkladať do výsledného C kódu vygenerovanej aplikácie. Jednoduchá ukážka časti TLC súboru s vygenerovaným C kódom je znázornená na Obr. 3.



Obr. 3 TLC súbor s vygenerovaným C kódom zo Simulinku

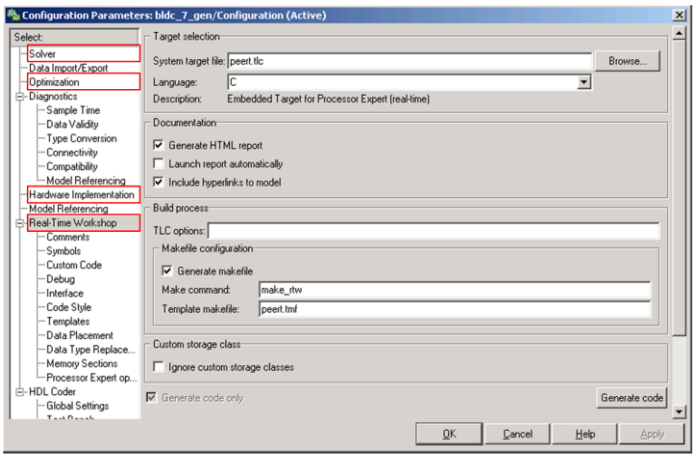
2.1.2.3 Nastavenie RTW

Ak chceme vygenerovať zdrojové kódy Simulinkového modelu, je nutné nastaviť potrebné parametre v dialógovom okne „Configuration Parameters”, ktoré je znázornené na Obr. 4. Nastavenia v záložkách *Solver*, *Optimalization*, *Hardware Implementation* a *Real-Time Workshop* môžu ovplyvniť chovanie simulinkového modelu, pri simulácií a generovaní kódu prostredníctvom RTW. Podrobnú diskusiu ohľadom nastavenia parametrov je možné nájsť v dokumente k RTW [2].

V rámci diplomovej práce bol použitý RTW target PEERT s nastaveniami parametrov:

- *Solver*: Pri generovaní kódu je nutné nastaviť typ solveru v danej záložke na „Fixed-step“ s odpovedajúcou periódou vzorkovania. Pri simulácií môže byť použitý aj Variable-step solver.
- *Optimization*: Nastavenia v danej záložke ovplyvňujú výkonnosť generovaného kódu. Oproti pôvodnému nastaveniu som nevyužil funkciu „Block reduction“, ktorá odstraňuje nepotrebné bloky pri generovaní kódu zo Simulinkového modelu. Docielil som tým možnosť používať bloky „Unit Dealy“ ako pamäťové miesta na cieľovom HW k ukladaniu veličín určených k vizualizácií.
- *Hardware Implementation*: definuje cieľový HW. Použité nastavenia:„Device type: 16 bit Generic Embedded Processor”, „Byte Ordering: Big Endian”, „Signed integer division rounds to: Undefined”

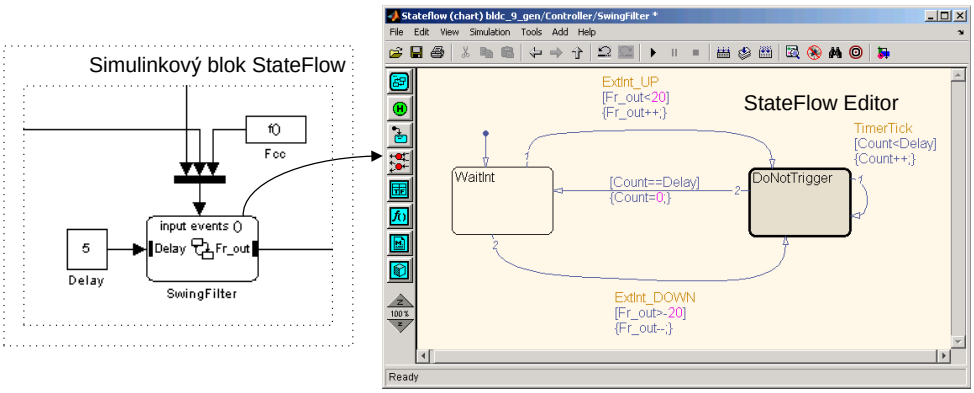
- *Real-Time Workshop*: Nastavenia záložky sú znázornené na Obr. 4. Podstatné je nastavenie cieľového Targetu, „System target file:peert.tlc" a jazyku automaticky generovaného kódu „Language: C".



Obr. 4 Dialógové okno Configuration parameters

2.1.3 StateFlow

StateFlow [4] je interaktívny vývojový nástroj určený k modelovaniu udalosťami riadených systémov. K implementácií daných systémov sa používa teória konečných automatov. S pomocou StateFlow grafu je možné doplniť Simulinkový model o zložitú, udalosťami riadenú logiku, z ktorej sa dá automaticky generovať C kód zo Simulinku, prostredníctvom RTW. StateFlow grafy sú reprezentované skupinami stavov, prechodov a parametrov ktoré môžu mat vstupný, výstupný, poprípade lokálny charakter. Prechody medzi stavmi môžu byť povoľované buď pomocou externých udalostí (event), poprípade prostredníctvom doplňujúcich podmienok.



Obr. 5 StateFlow editor

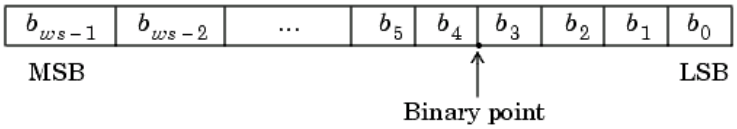
Stručný postup vytvárania automatu v StateFlow editore:

- vyberieme a vložíme StateFlow bloku do simulinkovej schémy.
- otvoríme StateFlow blok pomocou Model Exploreru.
- zadefinujeme vstupné, výstupné, lokálne premenné a udalosti vstupujúce do bloku (event).
- pridáme a pomenujeme stavy konečného automatu.
- vložíme obecný prechod (Default Transition) ktorý definuje počiatočný stav automatu.
- prepojíme stavy automatu pomocou prechodov.
- nastavíme podmienky prechodov:
    - Prechody povoľované prostredníctvom udalostí (event) sa určujú pridaním názvu udalosti do textového poľa k prechodu. Pr.: *ExtInt_DOWN*.
    - Doplňujúca podmienka prechodu medzi stavmi sa vkladá do textového poľa prechodu, do hranatých zátvoriek. Pr.: *[Fr_out>20]*.
- zadefinujeme výpočty a akcie StateFlow automatu:
    - Výpočty vykonávané pri prechode medzi stavmi je možné zapísať do textového poľa prechodu, do zložených zátvoriek. Pr.: *{Fr_out++;}*.
    - Výpočty vykonávané pri zotrvaní, príchode a odchode zo stavu automatu je možné zadefinovať s pomocou rezervovaných výrazov vpísaných do textového poľa daného stavu. Pr.: *Entry: akcia; During: akcia; Exit: akcia;*

2.1.4 Fixed-Point settings

Nástroj Fixed-Point settings rozširuje schopnosti Simulinku pri simulovaní a nastavovaní parametrov systémov, ktorých výpočtová časť je založená na fixed-point aritmetike. Fixed-point aritmetika pracuje s číslami, so zadefinovanou fixnou pozíciou binárnej čiarky, ktorá rozdeľuje binárne číslo na celú a zlomkovú časť. Obecné fixed-point číslo je zobrazené na Obr. 6, kde MSB predstavuje najvýznamnejší bit binárneho čísla,

LSB najmenej významný bit binárneho čísla, Binary point určuje pozíciu binárnej čiarky, WS určuje dĺžku binárneho čísla.



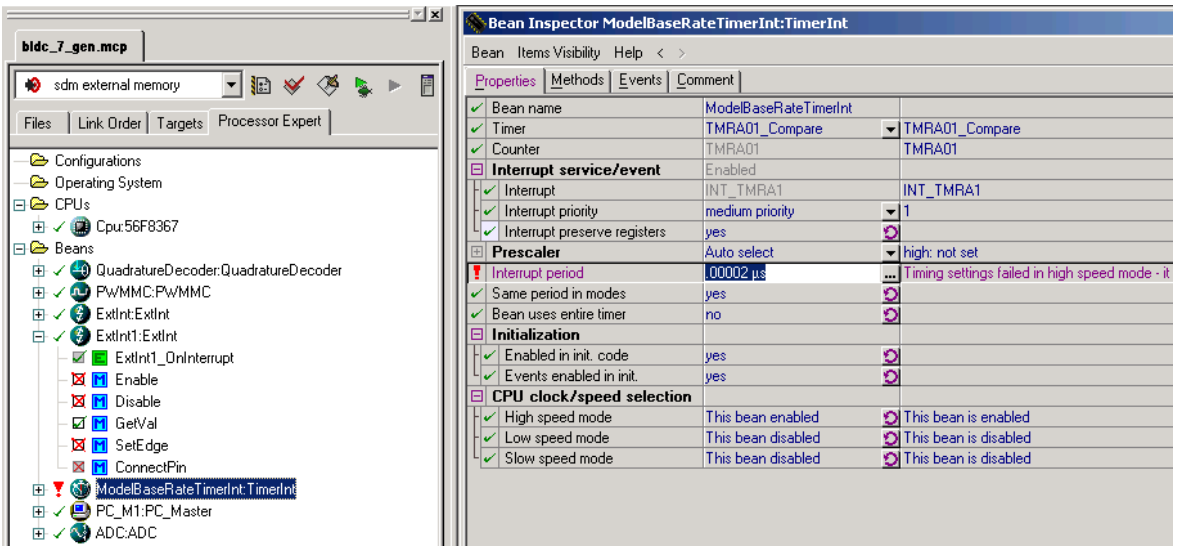
Obr. 6 Obecné fixed-point číslo [19]

Hodnota 16 bitového Unsigned Fixed-point Integer, s binárnou čiarkou rozdeľujúcou premennú na 8 bitov celej a zlomkovej časti, má maximálnu hodnotu $V = 2^{-8}.(2^{16} - 1) = 255{,}9961$ s rozlišovacou schopnosťou $\Delta V = 0{,}0039$. Pre 16 bit Signed Fixed-point Iteger by daná pozícia binárnej čiarky určovala hodnotu v rozmedzí $V = \langle -128 \quad +127.9961 \rangle$ s rozlišovacou schopnosťou $\Delta V = 0{,}0039$.

Nastroj Fixed-Point nám uľahčuje prácu so systémami s fixed-point aritmetikou, pri nastavovaní rozlišovacích schopnosti používaných premenných a výpočtov. Fixed-Point nástroj umožňuje previesť simuláciu simulinkovéj schémy, so záznamom maxima a minima jednotlivých výpočtov. Po skončení simulácie sa môžu automaticky aplikovať nastavenia jednotlivých výpočtových blokov tak, aby poskytovali maximálnu rozlišovaciu schopnosť bez rizika pretečenia pri aritmetických operáciách. Bližšie informácie k používaniu nástroja Fixes-Point je možné nájsť na [19].

2.2 Processor Expert

Processor Expert (PE) [9] je nástroj Rapid Aplication Designu (RAD) vyvíjaný spoločnosťou UNIS, ktorý slúži ako nadstavba vývojovému prostrediu Code Warrior (CW) [10]. Processor Expert zjednodušuje a unifikuje prístup k tvorbe SW časti embedded zariadení založených na rodine procesorov firmy Freescale. S PE je práca pokrývajúca úlohu s písaním inicializačného C kódu embedded zariadenia, predstavovaná len nastavovaním parametrov v grafickom užívateľskom prostredí (GUI). PE využíva prostredie CW k editovaniu a prekladaniu zdrojových kódov výslednej aplikácie, s možnosťou programovania cieľového HW.



Obr. 7 Processor Expert projekt s použitými Embedde Beans

PE obsahuje skupinu blokov, takzvaných *Embedded Beans* (EB) ktoré v sebe uchovávajú informácie o HW a SW možnostiach podporovaných procesorov a ich periférií. EB môžu predstavovať cieľový procesor, jednotlivé on-chip periférie procesora, externé periférie procesora, virtuálne zariadenia a čisto SW algoritmy. Spomenuté Embedded Beany majú zadefinované v rámci svojej štruktúry parametre, metódy a udalosti, ktoré sa môžu používať pri programovaní SW časti Embedded zariadenia, ako pri objektov orientovanom programovaní (OOP). Parametre jednotlivých EB určujú ich nastavenia , poskytované metódy slúžia k ovládaniu EB pri volaní z kódu aplikácie a udalosti sa využívajú k ošetreniu prerušení procesora od udalosti vyvolanej EB. Parametre EB sa môžu nastavovať v rozmedzí povolených hodnôt. V prípade nesprávneho nastavenia Embedded Beanov je chyba signalizovaná Processor Expertom. Ukážka je znázornené na Obr. 7, obsahujúceho Processor Expert projekt s použitými EB .

Veľkou výhodou využívania PE je prenositeľnosť tvorenej aplikácie medzi rôznymi druhmi podporovaných mikroprocesorov. Pri potrebe zmeny cieľového procesora aplikácie, stačí jednoducho pridať EB nového cieľového procesora do existujúceho projektu PE. Po zmodifikovaní nastavení HW periférií pre nový procesor, stačí zavolať spracovanie zmien príkazom Make a výsledná aplikácie je hotová k nahrávaniu do cieľového HW.

Skupiny Embedded Beanov obsiahnutých v PE:

- CPU: skupina obsahuje Beany podporovaných cieľových procesorov PE.
- CPU External Devices: skupina obsahuje Beany ako SW emulované zbernice I2C, SPI, Beany pre komunikáciu po zbernici RS-232, externé I/O moduly ...
- CPU Internal Devices: obsahujú Beany reprezentujúce: A/D prevodníky, I/O porty, Externé prerušenia, Časovače, Čítače ...
- SW: obsahuje širokú škálu SW Beanov pre riadenie motorov, správu pamäte, dát, bezpečnostné knižnice, matematické knižnice....

2.3 Peslib a PEERT

Nástroj integrujúci prostredie Processor Expert do prostredia Matlab/Simulink sa skladá z troch častí:

Prvá časť knižnica Peslib, zabezpečuje prepojenie prostredia Processor Expert, s prostredím Simulink prostredníctvom PE COM serveru. Prepojenie synchronizuje simulinkový model a jeho bloky s odpovedajúcim projektom v PE.

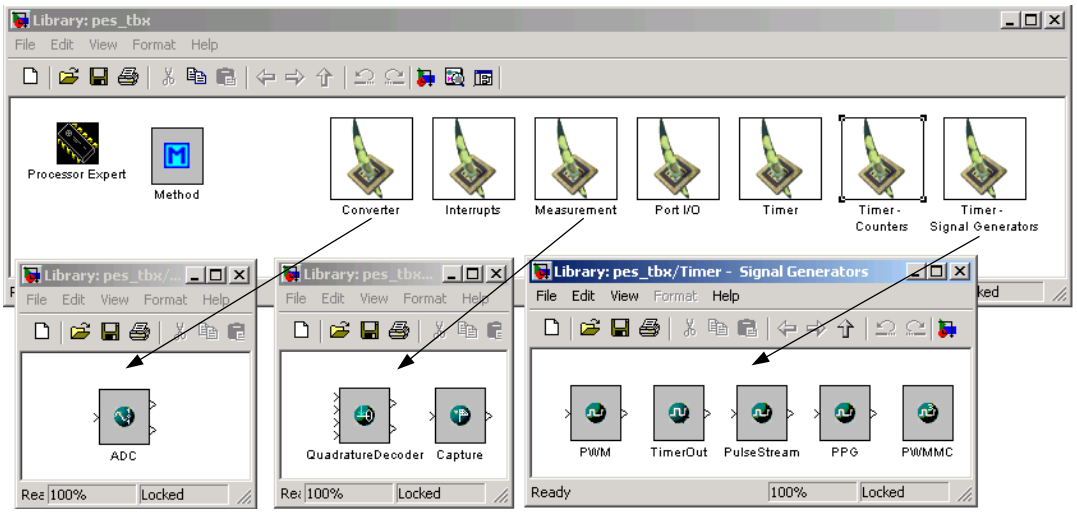
Druhá časť, Peslib blockset, obsahuje knižnicu simulačných blokov ktoré reprezentujú EB použité v PE. Spomenutý block set slúži k simulácií chovania HW periférií mikroprocesorov, ktorých parametre nastavujeme v PE. Podrobnejšie rozobratie podporovaných HW blokov je v nasledujúcom odstavci 2.3.1.

Tretiu časť knižnice, Processor Expert Embedded Real Time Target (PEERT) tvorí Embedded Real Time Workshop Target (ERT), ktorý vykonáva funkciu automatického generovania zdrojového kódu simulinkového modelu.

Podrobnejšie informácie k Peslib a PEERT sú dostupné v dokumente [5].

2.3.1 Podpora HW periférií knižnicou Peslib

Ako už bolo spomenuté, Peslib knižnica obsahuje block set reprezentujúci vybrané periférie mikroprocesorov. Ukážka Peslib block setu je na Obr. 8.



Obr. 8 Peslib block set

Peslib block set obsahuje tri druhy blokov:

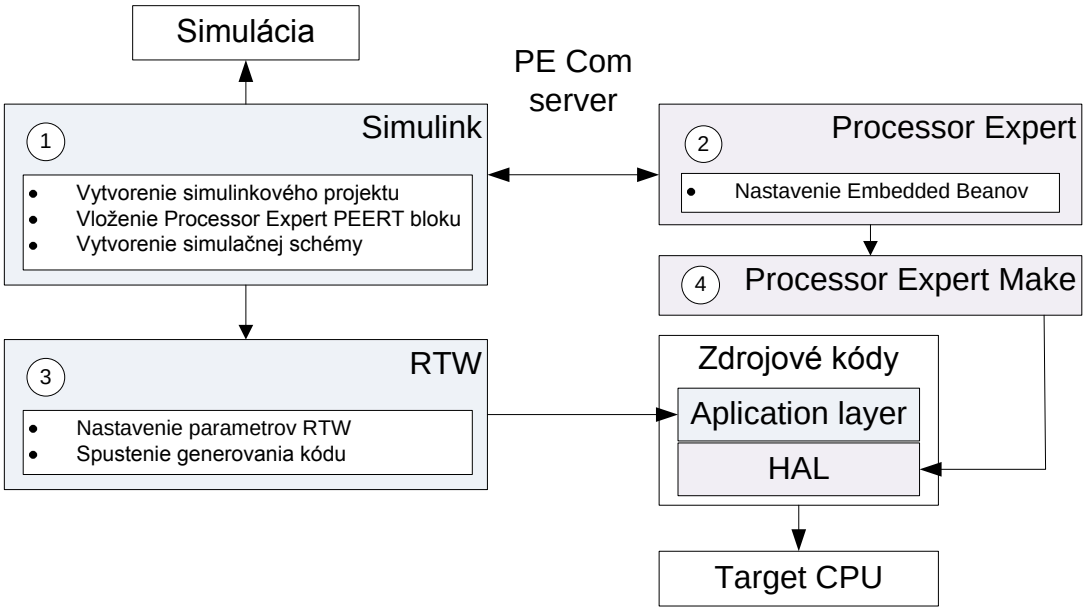
- Blok s názvom *Processor Expert* sa pridáva do simulinkového projektu pri vytvorení modelu v Simulinku. Daný blok nadviaže spojenie medzi Simulinkom a Processor Expertom prostredníctvom PE COM serveru [5]. Aby bolo prepojenie projektu v Simulinku a PE úspešné, musí sa daný blok pridať do simulačnej schémy pred ostatnými blokmi Peslib knižnice.
- Bloky reprezentujúce štandardné periférie mikroprocesora, ako napríklad AD prevodníky, časovače, čítače, I/O porty, generátory PWM signálov. Bloky slúžia k simulácií chovania jednotlivých HW periférií.
- Blok *Method* slúži k volaniu metód poskytovaných blokmi vložených do simulinkového projektu. Volanie metód jednotlivých Peslib blokov je možné v rámci simulačnej úlohy.

2.3.2 Tvorba projektu s využitím Peslib knižnice, generovanie kódu aplikácie.

Nasledujúci text poskytuje stručný opis postupu, akým sa vytvorí a nakonfiguruje projekt v Simulinku a PE s použitím Peslib knižnice.

Postup:

- Vytvoríme nový Simulink projekt, ktorý pomenujeme a uložíme do pracovného adresára.
- Pridáme do projektu Peslib blok Processor Expert, ktorým vytvoríme prepojenie medzi Simulinkom a PE. Po vložení sa vytvorí v pracovnom adresári podadresár s názvom modelu v ktorom sa ukladajú výsledky automatického generovania kódu zo Simulinkového modelu. V adresári sa vytvorí súbor s príponou **.pe*, ktorý predstavuje PE projekt.
- Vytvoríme Simulinkové schéma s použitím štandardných a Peslib blokov.
- Peslib bloky správne nakonfigurujeme prostredníctvom PE a prevedieme potrebné simulácie.
- Pri dosiahnutí požadovaných výsledkov simulácií, odstránime zo simulinkovej schémy nepotrebné časti, ktoré nechceme zahrnúť do automatickej generácie zdrojového kódu.
- Zmeníme nastavenia potrebných parametrov v dialógovom okne Configuration parameters spomenutých v odstavci 2.1.2.3.
- Spustíme automatické generovanie kódu zo Simulinku prostredníctvom RTW.
- Po úspešnom vykonaní procedúry generovania kódu, vyvoláme okno Processor Expertu, kde spustíme Make PE projektu. Interný Make PE spracuje vygenerovaný kód zo Simulinku a doplní ho o vlastnú vrstvu obsluhujúcu HW, takzvanú Hardware abstraction layer (HAL).
- Následne otvoríme existujúci projekt CodeWarrioru s názvom simulinkového projektu a príponou **.mcp*. Zavoláme preklad zdrojových kódov, nahráme a spustíme projekt na cieľovom HW.



Obr. 9 Postup tvorenia aplikácie s využitím PEERT v Simulinku a PE

2.4 Vizualizačné prostredie FreeMaster

FreeMaster [17] je vizualizačné prostredie na strane PC, určené k real-time sledovaniu a ovládaniu embedded zariadení založených na mikroprocesoroch spoločnosti Freescale. Freemaster umožňuje komunikovať s pripojeným embedded zariadením cez zbernicu RS-232, poprípade s použitím rozhrania JTAG a LPT portu. K nadviazaniu úspešného spojenia s embedded zariadením je nutné vytvoriť v programe FreeMaster vizualizačný projekt, s načítanými adresami jednotlivých pamäťových miest sledovaných premenných. Na strane embedded zariadenia je nutné pridať v PE projekte, pred prekladom a nahrávaním do cieľového HW, bean PC_Master z kategórie CPU External Devices.

FreeMaster umožňuje vizualizáciu dát v textovej a v grafickej forme s využitím softwarových osciloskopov (Obr. 53) a ovládacích panelov (Obr. 52). Ovládacie panely si môže užívateľ naprogramovať v tvare HTML stránky podporujúcej VBScript a JScript, používaných k čítaniu a prepisovaniu dát na sledovanom zariadení. Bližšie informácie k vytváraniu projektov v prostredí FreeMaster je možné nájsť v dokumentácií [17].

3 Rozšírenie podpory HW periférií Peslib knižnice

V rámci diplomovej práce bol rozšírený block set knižnice Peslib o simulačný blok HW periférie mikroprocesorov, PWMMC. Daný blok predstavuje trojfázový generátor PWM signálu, využívaný v Processor Experte primárne pri riadení trojfázových motorov.

K vytvoreniu PWMMC bloku bolo nutné naprogramovať S-funkciu, simulujúcu chovanie PWM generátora v základných parametroch a TLC súbor podporujúci automatické generovania C kódu zo Simulinku.

3.1 Implementácia S-funkcie PWMMC

Pri implementácií simulačného bloku PWMMC sme museli dodržať základné parametre HW periférie:

- Vstupom PWMMC beanu v PE je 16 bitový Unsigned Integer v rozsahu $u \in \langle 0,65535 \rangle$, ktorým nastavujeme šírku výstupného PWM pulzu. Stredná hodnota výstupného PWM signálu má hodnotu $\bar{y}_{PWM} = u/65535$.
- PWMMC produkuje PWM signál o frekvencií f, ktorého hodnoty môžeme meniť po zadefinovanom počte periód r v PE. Maximálna frekvencia nastavovania šírky výstupného pulzu PWMMC, nadobúda hodnotu f/r.

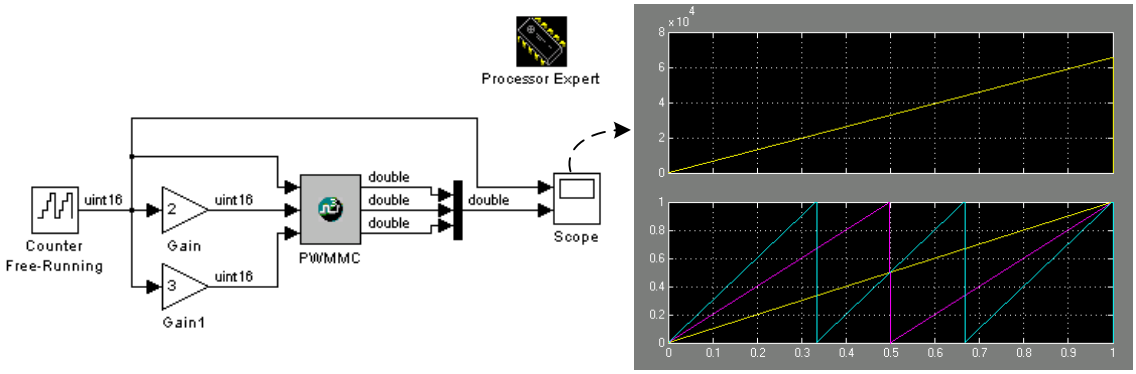
Implementácia S-funkcie, ktorá by simulovala chovanie trojfázového PWM generátora PWMMC, mohla mať dva druhy realizácie:

Prvá z možností, ktorá sa nám ponúkala, bola založená na podrobnej simulácií chovania PWMMC generátora, obsahujúca vnútorný čítač hodinových signálov, s komparátorom vstupnej hodnoty, ktorý by tvaroval výstupnú šírku pulzov generátora. Druhá z možností simulovania PWMMC generátora, ktorá sa nám ponúkala, je založená na počítaní strednej hodnoty výstupného PWM signálu. Výpočet výstupnej hodnoty PWMMC bloku by sa vykonával len v čase nastavovania novej hodnoty výstupu s frekvenciou f/r.

Prvá z variant by nám síce umožnila simuláciu chovania PWMMC generátora do podrobností, ale bola by neúnosne pomalá. K odsimulovaniu jednej periódy výstupného signálu by bolo potrebné vykonať 2^{16}-1 krokov. K tomu pripočítame výstupnú frekvenciu

PWMMC generátora nastavenú napríklad na 16 kHz a máme výpočtový problém ktorého vyriešenie by trvalo desiatky minút. Rozhodli sme sa preto pre implementovanie druhej varianty S-funkcie PWMMC bloku.

Samotná implementácia S-funkcie zahŕňala napísanie funkčnej časti volaných metód Simulinkom, opísaných v odstavci 2.1.1.1. Časť algoritmu S-funkcie prevádzajúcej výpočet vstupnej hodnoty bloku na výstup, je zdokumentovaný v Príloha B. Plné zdrojové kódy S-funkcie, sú k dispozícií na priloženom CD, v súbore pod názvom *pes_pwmmc.c*. Ukážka funkčnosti PWMMC bloku je znázornená na Obr. 10. Na ukážke je nastavovaný PWMMC blok vstupnou hodnotou 16 bitového Unsigned Integeru, ktorá sa prenáša na výstup PWMMC bloku vo forme strednej hodnoty PWM signálu o jednotkovej amplitúde.



Obr. 10 Ukážka funkčnosti PWMMC

3.2 Implementácia TLC súboru PWMMC

K zabezpečeniu podpory automatického generovania C kódu pre PWMMC blok, knižnice PEERT, bolo potrebné vytvoriť odpovedajúci TLC súbor pes_pwmmc.tlc. Obsah TLC súboru určuje, ktoré časti C kódu budú vkladané do výsledného zdrojového kódu aplikácie. Časť implementovaného TLC súboru pre blok PWMMC je vysvetlená v nasledujúcom texte. Plné zdrojové kódy TLC súboru sú k dispozícií na priloženom CD.

Časť zdrojového kódu TLC súboru pre blok PWMMC:

```
%function Outputs(block, system) Output
  %assign BeanName = block.Identifier

  %assign u1     = LibBlockInputSignal(0, "", "", 0)
  %assign u2     = LibBlockInputSignal(1, "", "", 0)
  %assign u3     = LibBlockInputSignal(2, "", "", 0)

  /* %<Type> Block: %<Name> */
    %<BeanName>_SetRatio16(0,%<u1>);
    %<BeanName>_SetRatio16(2,%<u2>);
    %<BeanName>_SetRatio16(4,%<u3>);
    %<BeanName>_Load();

  %if (NumSFcnSysOutputCalls > 0)
        %<CreateEventHandler (block, 0, "OnReload", TLC_TRUE, PurelyIntegerCode)>
  %endif
%endfunction
```

Vysvetlenie:

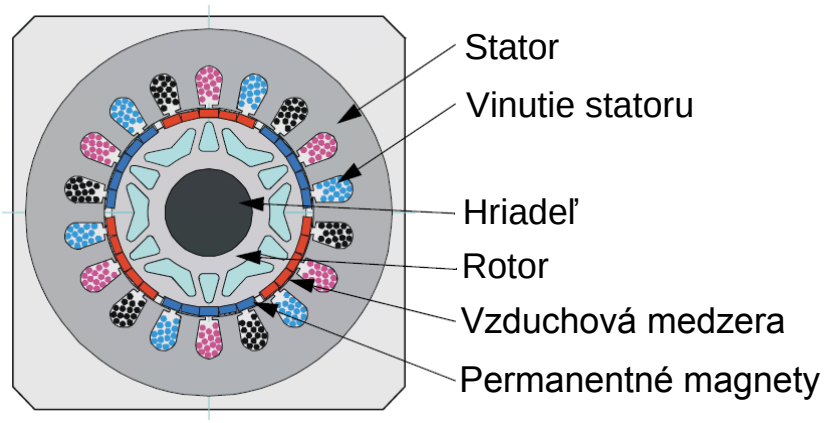
Ovládanie PWMMC bloku sa vykonáva vo finálnej verzií programu, volaním metód HAL [5], poskytovaných Processor Expertom. Nastavenie výstupnej hodnoty PWMMC sa vykonáva zavolaním metódy *<BeanName>_SetRatio16(byte číslo_kanála, uint16 perioda)*, pre každý z kanálov PWMMC generátora a zavolaním metódy *<BeanName>_Load()*, ktorá zabezpečí aplikovanie daných hodnôt na výstup PWM generátora. Parameter *<BeanName>* predstavuje pomenovanie bloku v Simulinku. TLC (Target Language Compiler) zabezpečí automatické vloženie kódu do výslednej formy s nahradením parametra *<BeanName>* za použitý názov bloku v simulinkovej schéme. Parameter vstupu *<ui>*, nahradí TLC odpovedajúcimi dátami vstupného signálu. Posledný príkaz TLC súboru *CreateEventHandler*, vygeneruje C kód ošetrujúci prerušenie procesora od príznaku *OnReload*, pri jeho prítomnosti. Proces vkladania kódu je vyobrazený na Obr. 3.

4 Popis demonštračnej úlohy

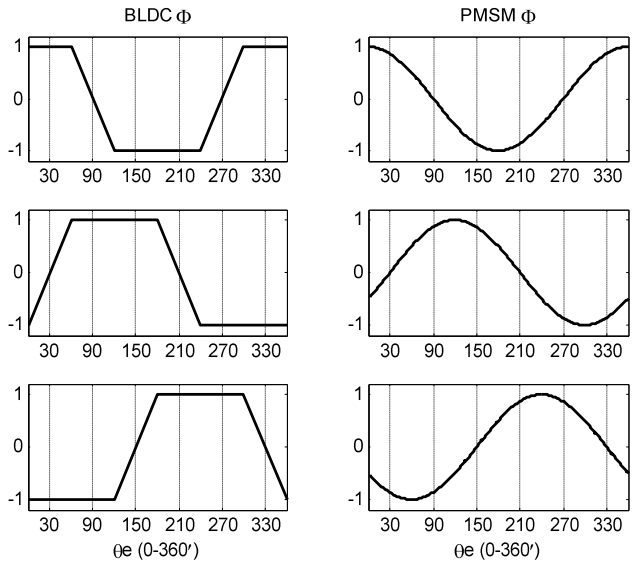
Hlavnou časťou diplomovej práce je ukážka aplikácie knižnice Peslib na demonštračnej úlohe. K diplomovej práci bola dodaná univerzálna vývojová doska MC56F8367EVM osadená DSP procesorom 58F8367 od firmy Freescale, ktorú sme programovali automaticky generovaným kódom zo Simulinku a Processor Expertu. Rozhodli sme sa preto využiť širokú škálu periférií poskytovaných osadeným procesorom a aplikovali sme ich na riadení trojfázového motora riadeného PWM. V rámci použitého HW prichádzali do úvahy dva druhy motorov: PMSM - Permanent magnet synchronous motor a BLDC – Brushless DC motor.

4.1 Porovnanie BLDC a PMSM motorov

PMSM a BLDC patria do skupiny synchrónnych trojfázových motorov. Ich rotor je zložený z permanentných magnetov a stator tvorí trojica budiacich cievok vzájomne posunutých o 120° el. Rozdiel vo fyzickej stavbe medzi týmito dvoma druhmi motorov je v spôsobe, akým majú navinuté a poprepájané statorové cievky. PMSM má zložitejšiu štruktúru vinutí než BLDC. Rozloženie spätnej indukcie elektromotorického napätia (BEMF) PMSM motora má sínusový tvar. BLDC motor má trapézové rozloženie BEMF. Rozdiely BEMF (back elektromotion force) sú znázornené na Obr. 12



Obr. 11 Rez BLDC motora [7]



Obr. 12 Rozloženie BEMF BLDC a PMSM motora [16]

Z mechanických rozdielov PMSM a BLDC vyplývajú aj rozdielne techniky riadenia motorov a ich aplikačné zamerania.

Hlavnou cieľovou oblasťou použitia PMSM sú presné servomechanizmy riadené na konštantné otáčky, prípadne polohu. Sínusové rozloženie BEMF skombinované s vektorovým riadením umožňuje ich plynulé polohovanie s konštantnou reguláciou krútiaceho momentu [7]. Nevýhodami PMSM motora sú vyššia nadobúdacia cena a vyššia výpočtová náročnosť riadiaceho algoritmu. Preto sa v štandardných aplikáciách, kde nie je potrebné precízne polohovanie, uprednostňuje BLDC motor.Vhodnou cieľovou oblasťou BLDC motorov sú aplikácie riadené na konštantnú frekvenciu otáčok. BLDC motor nie je vhodný k presnému polohovaniu rotora a to z dôvodu rozloženia BEMF. K ovládaniu BLDC motorov sa využíva metóda elektronického komutovania.

Nevýhody mechanickej konštrukcie BLDC sú kompenzované jednoduchosťou riadenia, ktorá je výpočtovo nenáročná. Rotujúce magnetické pole BLDC motor sa ovláda spínaním napätia vždy na dvoch vinutiach statora. Princíp komutácie je rovnaký ako pri mechanickom komutovaní DC motora. Rozdiel medzi BLDC a klasickým DC motorom je v absencií mechanického komutátora nahradeného riadiacim mikroprocesorom a prehodením funkčnosti statora s rotorom. Pri klasickom DC motore, stator tvoria permanentné magnety a rotor tvorí budiace magnetické pole vykonávajúce pohyb rotora.

Nahradením mechanického komutátoru elektronickým riadením, sa zvyšuje účinnosť (>90%), životnosť a znižuje hlučnosť motora.

K riadeniu polohovania a frekvencie otáčania PMSM a BLDC je potrebná spätná väzba od externých senzorov udávajúcich natočenie rotora. K získaniu natočenia rotora sa najčastejšie používajú inkrementálne senzory (IRC), prípadne informácia z Hallových sond u BLDC. Sú vypracované aj techniky bez senzorového riadenia BLDC a PMSM motora. Bezsenzorovým riadením motorov sa v tejto práci zaoberať nebudeme.

V rámci diplomovej práce bola vykonaná počiatočná implementácia riadenia na PMSM a BLDC motore. Postupom času sme sa priklonili k riešeniu riadenia BLDC motora z dôvodu dostupnosti HW. V ďalšej práci bude rozobratá aplikácia riadenia BLDC motora s využitím automaticky generovaného kódu zo Simulinku a Processor Expertu. Simulinkový model PMSM motora ktorý bol vytvorený v úvodnej fáze diplomovej práce, je zdokumentovaný v Príloha A

4.1.1 Matematický popis BLDC motora

Matematický popis BLDC motora [19] sa dá rozdeliť na elektrickú a mechanickú časť.

Rovnica 1 Popis elektrickej časti BLDC motora:

$$\begin{aligned}
\frac{d}{dt}i_a &= \frac{1}{3L_S}(2v_{ab} + v_{bc} - 3R_s i_a + \lambda p\omega_r(-2\Phi'_a + \Phi'_b + \Phi'_c)) \\
\frac{d}{dt}i_b &= \frac{1}{3L_S}(-v_{ab} + v_{bc} - 3R_s i_b + \lambda p\omega_r(\Phi'_a - 2\Phi'_b + \Phi'_c)) \\
\frac{d}{dt}i_c &= -\frac{d}{dt}i_a - \frac{d}{dt}i_b \\
T_e &= p\lambda(\Phi'_a \cdot i_a + \Phi'_b \cdot i_b + \Phi'_c \cdot i_c)
\end{aligned}$$

Kde :

i_a, i_b, i_c	prúdy tečúce jednotlivými cievkami statorového vinutia
v_{ab}, v_{bc}	rozdielové napätia aplikované na jednotlivých cievkach statorového vinutia
R_S	odpor cievky statorového vinutia
L_S	indukčnosť cievky statorového vinutia

$\Phi'_a, \Phi'_b, \Phi'_c$	rozloženie spätnej indukcie elektromotorického napätia rotorom do statorového vinutia (BEMF)
λ	magnetický toku permanentného magnetu
ω_r	uhlová rýchlosť rotora
p	počet pólových párov vinutí motora
T_e	elektromagnetický krútiaci moment

Rovnice elektrickej časti motora nám popisujú chovanie zmien prúdov tečúcich vinutiami cievok na aplikovanom svorkovom napätí, odčítanú o hodnotu napätia z parazitných odporov cievok a napätia spätne indukovaného rotorom motora. Z dôvodu zapojenia cievok BLDC motora do hviezdy stačí k popisu tretieho prúdu vzťah $i_c = -i_a - i_b$ (1. Kirchhoffov zákon). Časová konštanta nábehu prúdu cievky pri zapojení napätia nadobúda hodnotu $\tau = L/R$. Táto hodnota je dôležitá pre nastaveniu spínacej frekvencie PWM generátora. Rozloženie BEMF je funkciou elektrického uhlu natočenia rotora $\Phi'_{a,b,c} = f(\theta_e)$. Priebeh BEMF je znázornený na Obr. 12.

Hodnota elektromagnetického krútiaceho momentu závisí na počte pólových dvojíc motora, na hodnote magnetického toku rotora, jeho polohe a prúdov tečúcich cievkami statora.

Rovnica 2 Popis Mechanickej časti BLDC motora

$$\frac{d}{dt}\omega_r = \frac{1}{J}(T_e - F\omega_r - T_m)$$
$$\frac{d\theta}{dt} = \omega_r$$
$$\theta_e = p \cdot \theta$$

Kde :

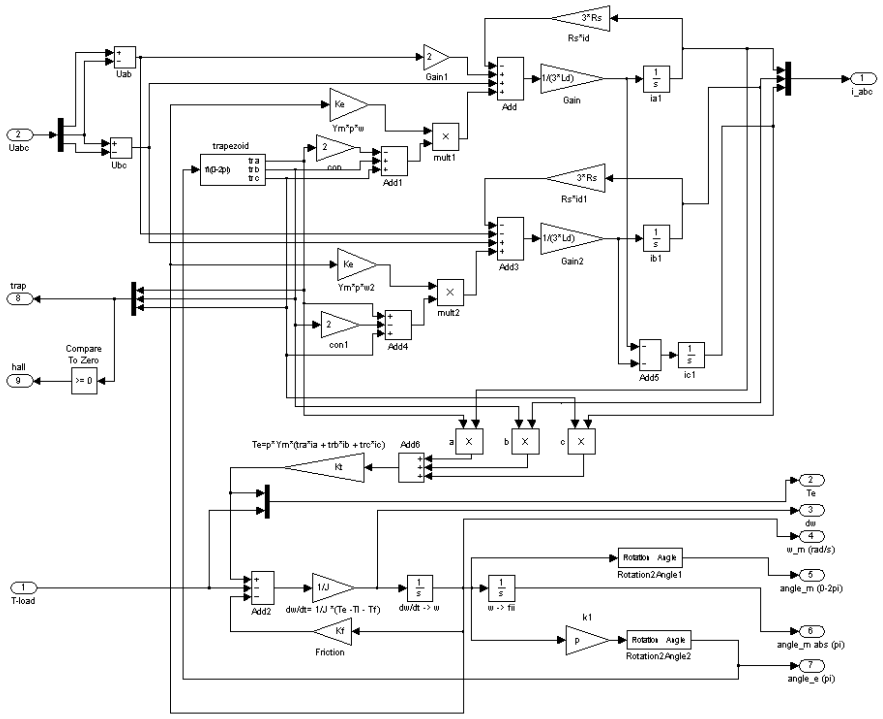
ω_r	uhľová rýchlosť rotora
θ	uhol natočenia rotora
θ_e	uhol elektrického natočenia rotora
J	kombinovaná zotrvačnosť statora a záťaže

p	počet pólových párov vinutí motora
T_e	elektromagnetický krútiaci moment
T_m	mechanický krútiaci moment záťaže
F	viskózne trenie

Popis mechanickej časti nám podáva informáciu o dynamických vlastnostiach motora. Derivácia uhlovej rýchlosti rotora je priamo úmerná na krútiacom momente vyvolanom cievkami statora, záťažou a viskóznym trením. Derivácia uhlovej rýchlosti je nepriamo úmerná kombinovanej zotrvačnosti rotora a záťaže. Zmena uhlu natočenia rotora je závislá na uhlovej rýchlosti rotora.

4.1.2 Simulačná schéma BLDC motora

K simulačným účelom bol vytvorený nelineárny simulinkový model BLDC motora, vyobrazený na Obr. 13. Model je presnou interpretáciou matematického popisu. Vstupnými a výstupnými veličinami sú všetky premenné parametre spomenuté v predošlom odstavci, s doplnením o výstup Hallových sond. V rámci diplomovej práce bol vytvorený aj simulinkový model PMSM motora, ktorý je zdokumentovaný v Príloha A. Pri simulácií boli použité reálne parametre motora MCG IB-23811, ktorý slúžil ako cieľový HW určený k riadeniu. Fyzikálne parametre motora používané v simulácií sú uvedené v Tab. 1.



Obr. 13 Simulačný model BLDC motora

Tab. 1: Parametre motora MCG IB-23811

Charakteristika	Konštanta	Hodnota	Jednotka
Počet pólových dvojíc	p	2	-
Maximálne svorkové napätie	Udc max	170	V
Maximálny trvalý prúd vinutím	Idc max	1,8	A
Indukčnosť cievky vinutia	L	6,8	mH
Odpor cievky vinutia	R	2,05	Ω
Konštanta krútiaceho momentu	Kt	0,084	Nm/A
Konštanta spätnej indukcie napätia	Ke	0,084	V /rad/s
Zotrvačnosť rotora	J	0,00013	g/m^2
Konštanta trenia	Kf	0,00005	Nm/rad/s

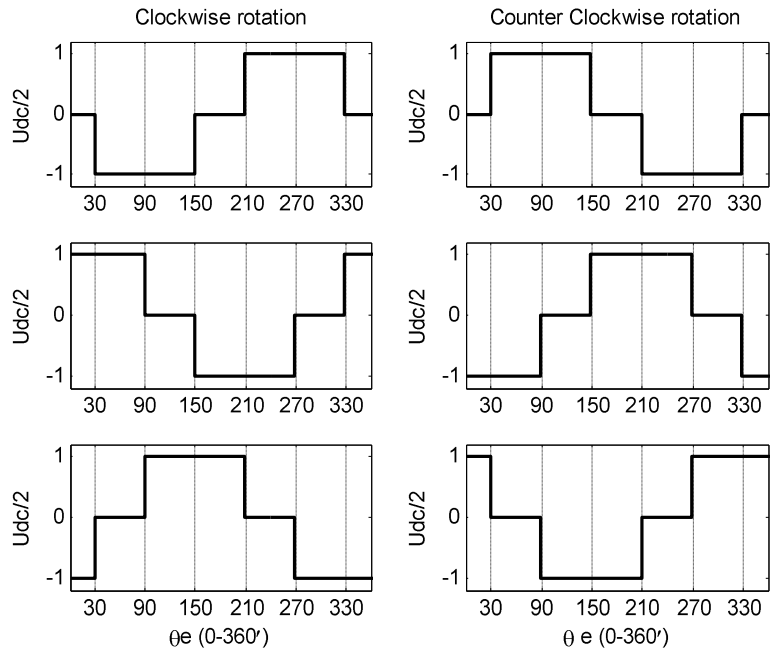
4.2 Digitálne riadenie BLDC motora

Ako už bolo spomenuté v odstavci 4.1, BLDC motor sa riadi pripájaním napätí vždy k dvom cievkam statora (komutácia). Kombinácie cievok na ktoré sa pripájajú dané napätia sú závislé na natočení magnetického poľa rotora, k pozícií jednotlivých vinutí statora. Aby bol efekt elektrického krútiaceho momentu čo najväčší, snažíme sa vytvárať magnetické pole statora posunuté približne o 90° el. od magnetického poľa permanentných magnetov rotora. Ukážka tvorby rotujúceho magnetického poľa statora pripájaní

jednotlivých svorkových napätí je znázornená na Obr. 14. Na ukážke je zobrazená aplikácia maximálnych amplitúd napätí pripájaných k BLDC.

K napájaniu BLDC motorov sa používa trojfázový H-mostík (Obr. 23), ktorého výstupné napätie je pulzne šírkovo modulované (PWM). Trojfázový H-mostík sa skladá zo šiestich spínacích prvkov, ktoré môžu produkovať šesť kombinácií napätí, vhodných pre napájanie motora. Takéto riadenie sa nazýva Six-Step Control [7]. H-mostík by mal generovať PWM pulzy len na výstupoch, ktorých napätia sú nastavované k vytváraniu elektromagnetického poľa rotora. Generátor PWM pulzov by mal podporovať maskovanie jednotlivých výstupov.

Ak je frekvencia spínania výstupného PWM dostatočne vysoká, BLDC motor reaguje na strednú hodnotu aplikovaného napätia PWM generátora, čomu odpovedajú aj prúdy tečúce cievkami statora a výsledný krútiaci elektromagnetický moment.



Obr. 14 Aplikované svorkové napätia pri tvorbe rotujúceho EM poľa statora

4.2.1 Meranie polohy rotora BLDC

Jednou z dôležitých súčastí pri riadení BLDC je stanovenie presnej polohy rotora a to z dôvodu určovania intervalov komutácie. BLDC motory majú väčšinou vo svojej vnútornej štruktúre zabudované senzory na bázy Hallových sond, poprípade namontované externé IRC senzory pripojené ku Quadrature dekodéru. Rozdiel v používaní medzi týmito dvoma senzormi je hlavne v tom, že na rozdiel od Hall senzorov nám IRC senzory s Quadrature dekóderom nepodávajú informáciu o absolútnej polohe rotora. Z toho dôvodu je nutné pri inicializácií riadiaceho algoritmu vykonať štartovacie procedúry nastavajúce Quadrature dekodér na správnu, absolútnu hodnotu polohy rotora.

Hodnoty Hall senzorov môžeme sledovať s využitím Peslib beanov zo skupiny Port I/O, konkrétne Bits I/O. Vyvolanie prerušenia procesora od Hall senzorov môžeme dosiahnuť použitím ExtInt beanu zo skupiny Interupts. Hodnoty Quadrature Dekodéru je možné získavať priamo z Peslib beanu QuadratureDecoder, zo skupiny Measurement.

4.2.1.1 Hallové senzory

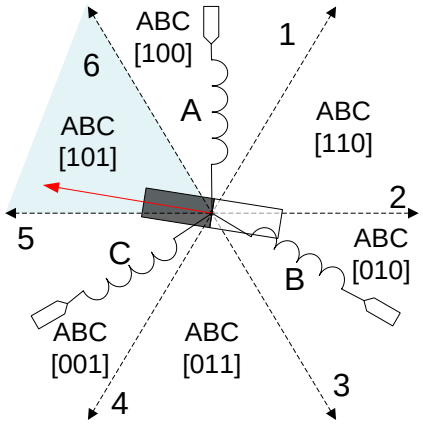
Hallové senzory (HS) zabudované do tela motora, nám poskytujú informáciu o absolútnom natočení vektora magnetického poľa rotora k rozmiestneniu statorových vinutí. HS rozdeľujú priestor jednej elektrickej otáčky rotora na šesť sektorov a identifikujú, v ktorom z nich sa daný vektor magnetického poľa nachádza. Ukážka rozdelenia jednotlivých sektorov s hodnotami HS je na Obr. 15. Výstupná hodnota HS je v digitálnej forme. Výhodou použitia HS je ich schopnosť určiť polohu natočenia rotora aj po výpadku napájacieho napätia.

Výhody Hallových senzorov:

- Výstupom je hodnota absolútneho natočenia rotora k cievkam statora.
- Výstup Hallových senzorov určuje priamo okamih komutácie.

Nevýhody Hallových senzorov:

- Nízka rozlišovacia schopnosť Hall senzora, ktorá nadobúda hodnotu $60°/p$, kde p predstavuje počet pólových dvojíc rotora.



Obr. 15 Identifikácia polohy rotora Hall senzormi

4.2.1.2 IRC senzor s Quadrature dekodérom

IRC senzory (Incremental radial senzor) patria do skupiny optických rotačných senzorov, ktoré fungujú na princípe generovania obdĺžnikových napäťových pulzov posunutých o štvrť periódy. Pulzy sú tvorené osvecovaním kódovacieho terčíku zdrojom svetla a ich detekciou pomocou fotodiód. IRC senzory majú väčšinou dva kanály výstupných pulzov, channel A a B. Mechanická rozlišovacia schopnosť IRC senzorov je daná počtom priechodov svetla na jeden kanál. V demonštračnej úlohe bol použitý IRC senzor firmy Agilent Technologies HEDS-5640-A06 [11] s 500 pulzmi na jednu otáčku na kanál. Výsledná rozlišovacia schopnosť senzora je 4x500=2000 pozícií na jednu mechanickú otáčku rotora. Okrem dvoch kanálov výstupných pulzov, IRC senzor poskytuje aj tretí výstup, Index channel. Index channel sa používa k určovaniu referenčnej polohy IRC senzora (jeden pulz na celú mechanickú otáčku). Použitý IRC senzor môže merať frekvenciu otáčania rotora do 500 Hz.

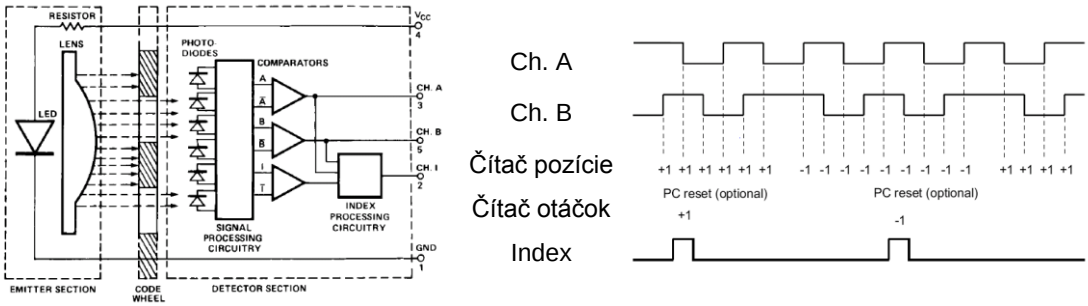
Samotný IRC senzor nám nepodáva informáciu o polohe rotora. K získaniu polohy rotora musíme výstupné priebehy IRC senzora dekódovať a počítať Quadrature dekodérom (QD). QD dekodér je implementovaný na HW úrovni v rámci samotného DSP osadeného na vývojovej doske. QD dekodér inkrementuje alebo dekrementuje hodnotu vnútorných registrov procesora pri zmene logického stavu A a B kanála. Výstup signálu Index sa najčastejšie používa k resetovaniu registrov QD, poprípade k počítaniu mechanických otáčok rotora. Princíp fungovania IRC a spôsob počítania výstupných pulzov QD dekodérom je znázornený na Obr. 16.

Výhody IRC senzorov s QD:

- Vysoká rozlišovacia schopnosť.
- Možnosť presného merania frekvencie rotora.

Nevýhody IRC senzorov s QD:

- Strata informácie o polohe rotora pri výpadku napájacieho napätia procesora.
- Nutný výpočet pozície v ktorej nastáva komutácia napätí.



Obr. 16 Princíp funkcie IRC senzora s QD dekodérom [11]

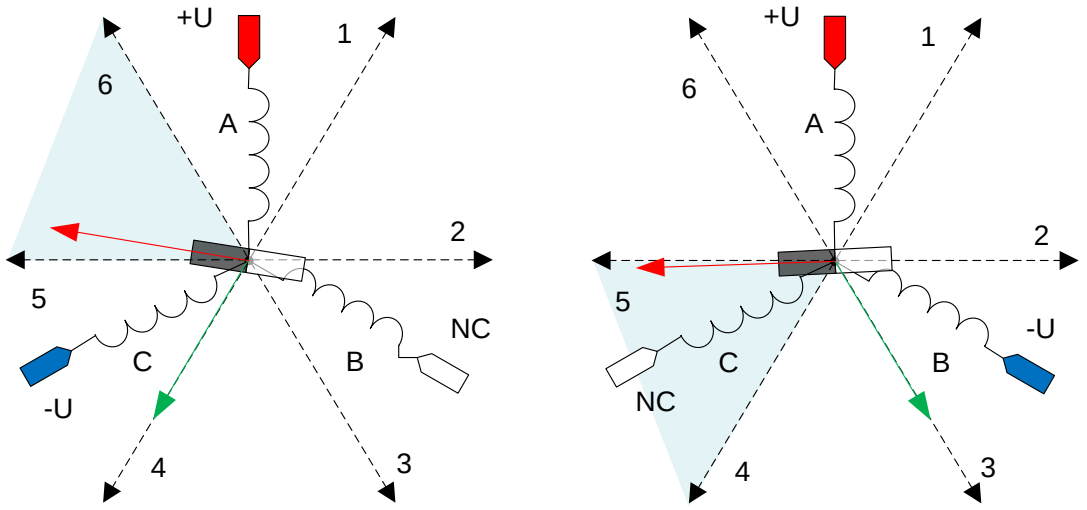
4.2.2 Komutácia BLDC motora

Komutáciou svorkových napätí statora vytvárame jeho rotujúce magnetické pole. Ako už bolo spomenuté v predošlom texte, k správnemu fungovaniu BLDC motora potrebujeme vytvárať elektromagnetické pole statora posunuté o 90° k vektoru magnetického poľa rotora. V nasledujúcom texte priblížime techniku komutácie napätí s využitím Six-step riadenie [7].

4.2.2.1 Komutácia BLDC s použitím Hall Senzorov

Na Obr. 17 je znázornená situácia pred komutáciou napätí BLDC motora. Vektor magnetického poľa statora sa nachádza v sektore 6. Poloha je identifikovaná Hall senzormi hodnotou *ABC [1 0 1]*. Pripojením kladného napätia na svorku vinutia A a záporného napätia na svorku vinutia C, vytvoríme elektromagnetické pole statora v smere 4. Spôsobíme rotáciu rotora proti smeru hodinových ručičiek. Keď vektor magnetického poľa rotora prejde do sektora 5, zmení sa výstup Hall senzorov na hodnotu *ABC [0 0 1]*. V tom momente nastáva komutácia. Svorkové napätie vinutia C sa odpojí a na svorku vinutia B sa pripojí záporné napätie. Výsledný vektor magnetického poľa statora vznikne v smere

3. Uhol medzi magnetickým poľom statora a rotora sa zvýši zo 60° na 120°. Rotor ďalej pokračuje v natáčaný do smeru elektromagnetického poľa statora. Prehľad kombinácií jednotlivých svorkových napätí pripájaných k BLDC motoru, pre jednotlivé hodnoty Hall senzorov, je zaznamenaný v Tab. 2.



Obr. 17 Stav pripojených svorkových napätí pred a po komutácii

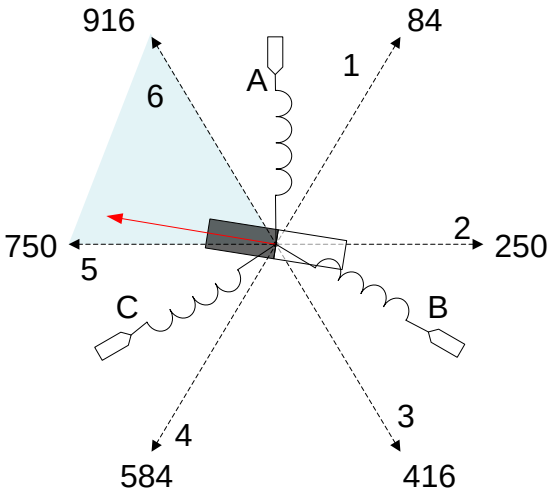
Tab. 2: Komutačné sekvencie BLDC motora

Sektor	Výstup Hallových senzorov			Pravotočivá rotácia			Ľavotočivá rotácia		
	A	B	C	fáza A	fáza B	fáza C	fáza A	fáza B	fáza C
1	1	0	0	NC	-U	+U	NC	+U	-U
2	1	1	0	+U	-U	NC	-U	+U	NC
3	0	1	0	+U	NC	-U	-U	NC	+U
4	0	1	1	NC	+U	-U	NC	-U	+U
5	0	0	1	-U	+U	NC	+U	-U	NC
6	1	0	1	-U	NC	+U	+U	NC	-U

4.2.2.2 Komutácia BLDC s použitím IRC a QD

Využívanie hodnoty Hallových senzorov pre komutáciu napätí je jednoduchšie ako pri hodnotách z QD dekodéra. Pri využívaní hodnôt QD dekodéra je nutné sledovať, v ktorej pozícií sa nachádza rotor a porovnávať túto hodnotu s prednastavenými hranicami komutácie. Je nutné určiť, v ktorom zo šiestich sektorov sa nachádza vektor magnetického poľa rotora. Príklad hraníc určujúcich komutáciu napätí pre motor s dvoma pólovými

dvojicami je na Obr. 18. Pre BLDC motor s dvoma pólovými dvojicami má jedna elektrická otáčka rotora maximálnu hodnotu QD registra 2000/2=1000. Je nutné z toho dôvodu nulovať QD registre pri presiahnutí tejto hodnoty.



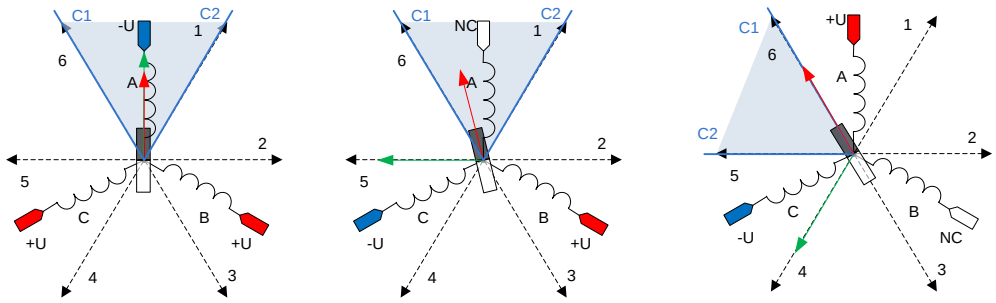
Obr. 18 Hranice komutácie s využitím QD

K určovaniu časového momentu komutácie s využitím hodnoty QD sa dá pristupovať dvoma spôsobmi.

Prvým spôsobom je cyklické porovnávanie hodnoty QD registrov s hranicami komutácie a zisťovaním, v ktorom zo sektorov elektrickej otáčky sa aktuálne rotor nachádza. Z tejto informácie sa už dá podľa Tab. 1 jednoducho zistiť kombinácia nastavovaných svorkových napätí motora.

Druhý spôsobom zisťovania časového momentu komutácie, je založený na vyvolaní prerušenia procesora od QD pri dosiahnutí hraníc komutácie. Postup techniky je znázornený na Obr. 19. V prvom kroku sa nastavia svorkové napätia vinutí tak, aby vektor magnetického poľa bol rovnobežný s niektorým z vinutí. Táto procedúra sa vykonáva len na začiatku behu programu, pri inicializácií. Keď sa rotor ustáli v známej pozícií, nastavia sa dva Compare registre QD na hodnoty, v ktorých bude vyvolané prerušenie procesora od QD. V druhom kroku sa spustí beh motora vytvorením magnetického poľa kolmého na pole permanentných magnetov rotora. V treťom kroku, sa pri prekročení hodnoty QD registra jednej z hraníc komutácie vyvolá prerušenie procesora. Vykoná sa komutácia svorkových napätí s následným nastavením nových hodnôt Compare registrov QD.

Prvý z prístupov je náročnejší na výpočtový čas procesora a to z dôvodu cyklického spúšťania procedúry komutácie motora. Frekvencia spúšťania procedúry komutácie pri druhom z prístupov je závislá na frekvencií otáčania rotora, tým pádom je menej náročná na výpočtový čas procesora.



Obr. 19 Nastavovanie Compare registrov QD pri detekcií komutácie

4.2.3 Meranie frekvencie otáčok BLDC rotora

4.2.3.1 Meranie frekvencie otáčok rotora s použitím IRC a QD

Výstupnou hodnotou QD je poloha rotora. Pri periodickom čítaní výstupu QD môžeme získať diferenciu polohy, z ktorej môžeme jednoducho podľa vzťahu Rovnica 3 vypočítať frekvenciu otáčok rotora. Frekvencia je závislá na perióde vzorkovania, na zmene polohy rotora a na rozlišovacej schopnosti IRC senzora. Pri tomto spôsobe merania frekvencie sa nevyužívajú vymoženosti HW, ako sú prerušenia. Tým pádom je meranie zaťažené väčšou chybou ako ostatné techniky merania frekvencie.

Rovnica 3 Výpočet frekvencie rotora pri cyklickom čítaní hodnoty QD

$$f = \frac{\Delta k}{T_{clk} \cdot N_1} \quad [Hz]$$

Rovnica 4 Absolútna chyba merania frekvencie pri cyklickom čítaní hodnoty QD

$$\varepsilon_f = \pm 0.25 \quad Hz \quad T_{clk} = 2ms \quad N_1 = 2000$$

Kde:

f mechanická frekvencia otáčania rotora

T_{clk} časová konštanta cyklického merania frekvencie

Δk diferencia QD registra od posledného čítania hodnoty

N_1 rozlišovacia schopnosť IRC senzora

Ďalšou možnosťou merania frekvencie je počítanie času jednej mechanickej otáčky rotora indikovanej signálom Index z IRC senzora. Prípadne meraním periódy výstupných signálov A a B z IRC senzora. K počítaniu času sa dá využiť Capture bean knižnice Peslib, ktorý obsahuje na pozadí bežiaci 16 bitový čítač. Pracovná frekvencia čítača môže byť až 60 Mhz, pre daný procesor 58F8367. Je možné nakonfigurovať Capture bean na vyvolanie prerušenia pri pretečení čítača, prípadne pri detekcií zmeny logickej hodnoty sledovaného signálu. Takéto meranie frekvencie sa vyznačuje vysokou presnosťou, ktorá závisí na hodinovom signáli počítaného čítačom.

Rovnica 5 Výpočet frekvencie rotora od signálu Index, IRC senzora

$$f = \frac{2^{16} - 1}{T_{MaxInt} \cdot k} \quad [Hz]$$

Rovnica 6 Výpočet frekvencie rotora od signálu A, IRC senzora

$$f = \frac{2^{16} - 1}{T_{MaxInt} \cdot k \cdot N_2} \quad [Hz]$$

Kde:

f výsledná frekvencia

T_{MaxInt} čas potrebný k načítaniu plnej hodnoty čítača

k hodnota registra Capture beanu pri vyvolaní prerušenia

N_2 počet pulzov kanála A IRC senzora

4.2.3.2 Meranie frekvencie otáčok rotora s použitím Hall senzorov

Princíp merania frekvencie otáčok rotora s Hall senzormi je rovnaký ako pri IRC senzoroch s QD. S využitím Capture beanu je možné sledovať zmeny logických hodnôt Hall senzorov a merať čas medzi jednotlivými komutáciami motora. Následne je už jednoduché spočítať výslednú frekvenciu rotácie rotora podľa vzťahu Rovnica 7.

Rovnica 7 Výpočet frekvencie rotora z časových intervalov komutácie

$$f = \frac{2^{16} - 1}{6p \cdot T_{MaxInt} \cdot k} \quad [Hz]$$

Kde:

f výsledná frekvencia

T_{MaxInt} čas potrebný k načítaniu plnej hodnoty čítača

k hodnota registra Capture beabu pri vyvolaní prerušenia

p počet pólových dvojí statora

4.2.4 Regulátor frekvencie otáčania rotora BLDC

Regulačná schéma BLDC motora je znázornené na Obr. 20. K regulovaniu otáčok BLDC bol použitý proporcionálno-integračný regulátor (PI) s obmedzením výstupu (Anti Wind-up). Výstupom z PI regulátora bola nastavovaná stredná hodnota svorkových napätí motora, prostredníctvom PWM generátoru. Trojfázový generátor PWM je v Peslib knižnici reprezentovaný blokom PWMMC. PWMMC bean reaguje na vstup v rozmedzí hodnôt 0-65535. Z toho dôvodu bolo nutné nastaviť anti Wind-up obmedzenie PI regulátora na hodnotu ±32767.

Rovnica 8 Popis PI regulátora v diskrétnej časovej oblasti

$$\begin{aligned} u(k) &= u_P(k) + u_I(k) \\ u_P(k) &= K_P \cdot e(k) \\ u_I(k) &= u_I(k-1) + K_I \cdot e(k) \end{aligned}$$

Kde:

$u(k)$ je výstup PI regulátora

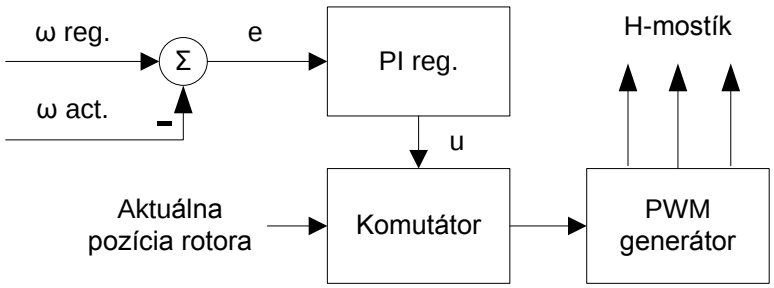
$u_P(k)$ je proporcionálna časť výstupu regulátora

$u_I(k)$ je integračná časť výstupu regulátora

K_P je proporcionálna konštanta regulátora

K_I je integračná konštanta regulátora

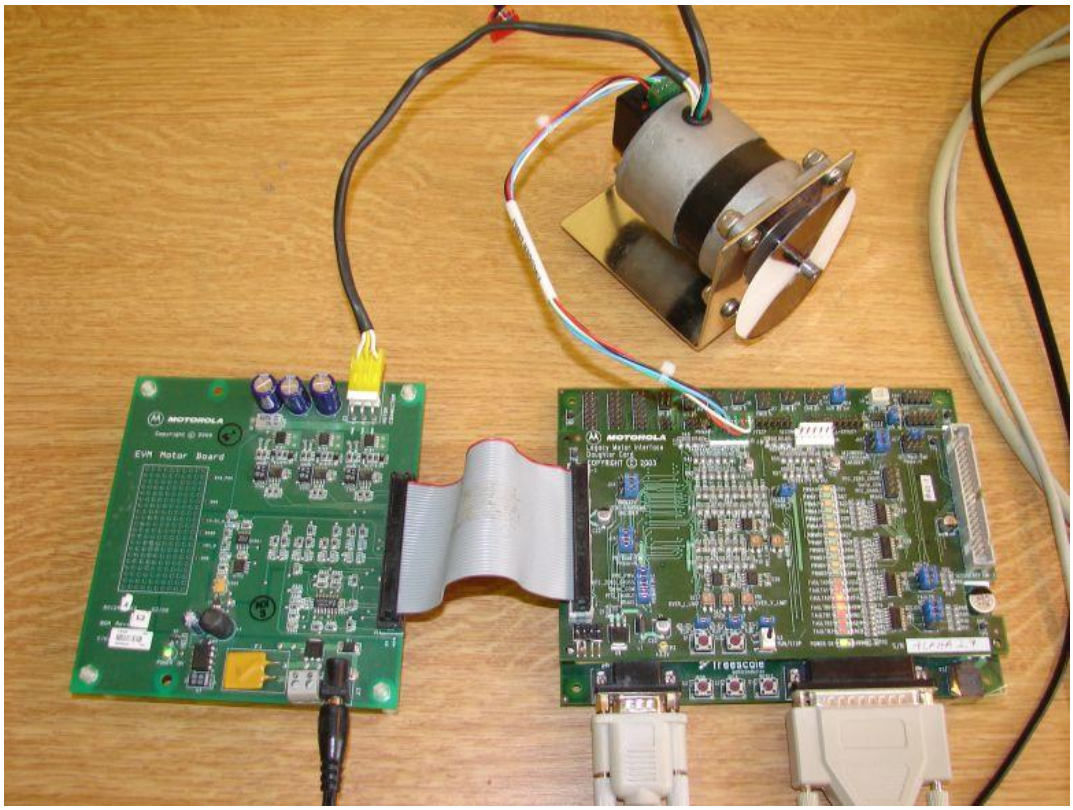
$e(k)$ je vstup PI regulátora



Obr. 20 PI regulátor frekvencie otáčanie rotora

4.3 Použitý HW

V rámci diplomovej práce bol použitý k testovaniu aplikácií HW, zložený z vývojovej dosky MC56F8367EVM [12] osadenej procesorom 56F8367 [8], nadstavbovej dosky Legacy Motor Interface Daughter Card [13], výkonovej časť EVM Motor Board [13] a motora MCG IB-23811 [14] s IRC senzorom HEDS-5640-A06 [11]. Podrobné špecifikácie jednotlivých častí HW je možné nájsť v priloženej dokumentácií. Fyzické nastavenie a zapojenie jednotlivého HW je zdokumentované v Príloha C.



Obr. 21 Vývojová doska MC56F8367EVM s výkonovou časťou a BLDC motorom

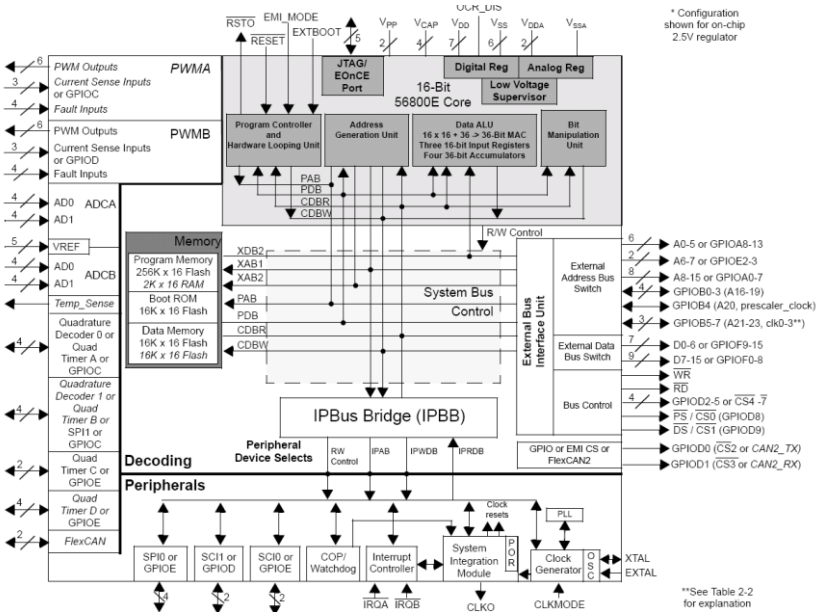
4.3.1 Vývojová doska MC56F8367EVM

Vývojová doska MC56F8367EVM [12] od firmy Freescale, je osadená digitálnym signálnym procesorom (DSP) 56F8367 [8], ktorého vnútorná štruktúra je znázornená na Obr. 22. Hlavnou výhodou procesora je jeho výpočtový výkon skombinovaný s množstvom I/O periférií. Hlavnými používanými perifériami v demonštračnej úlohe sú: PWM generátory, QD dekodér, AD prevodníky, Quad čítače. Procesor obsahuje aritmeticko-logickú jednotku (ALU) umožňujúcu prevádzať výpočty v 16 bit Signed Fixed-point Integer. Tento fakt treba brať do úvahy pri návrhu vnútornej štruktúry programu, pri nastavovaní rozlišovacej úrovne jednotlivých algoritmov. Vhodným nástrojom k simulovaniu a nastaveniu výpočtovej presnosti algoritmov v prostredí Simulink, je nástroj Fixed-Point.

Hlavné vlastnosti a poskytované periférie procesora 56F8367:

- Výpočtový výkon do 60 MIPS pri frekvencií procesorového jadra 60 MHz

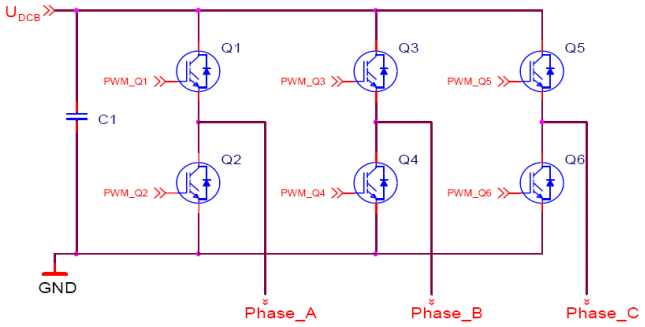
- Výkonné 16 bitové jadro rodiny 56800E založenom na duálnej Harvadr architektúre, simultánny prístup do troch miest pamäte: 1x programová, 2x dátová
- 16x16 bitová paralelná násobička (16x16 bit paralel Multiplier Accumulator MAC)
- Interná Programová pamäť: 512 KB Flash, 4 KB RAM, 32 KB BOOT ROM
- Interná Dátová pamäť: 32 KB Flash, 32 KB RAM
- 2x6 kanálov PWM
- 4x4 kanálov 12 bitových A/D prevodníkov
- 2x Quadrature dekodér (Quadrature Decoder)
- 4x Quad časovače (Quad Timer)
- 2x FlexCan moduly (vyhovujúce štandardu CAN 2.0 B)
- 2x SPI (Serial Peripheral Interface), 2xSCI (Serial Communication Interface)
- 2x vyhradený vstup externého prerušenia (IRQA, IRQB)
- 76x univerzálnych vstupno-výstupných vývodov procesora (General Purose I/O)
- JTAG/OnCE – rozhranie určené k programovaniu, neinváznemu testovaniu a debugovaniu aplikácií



Obr. 22 Bloková schéma 56F8367

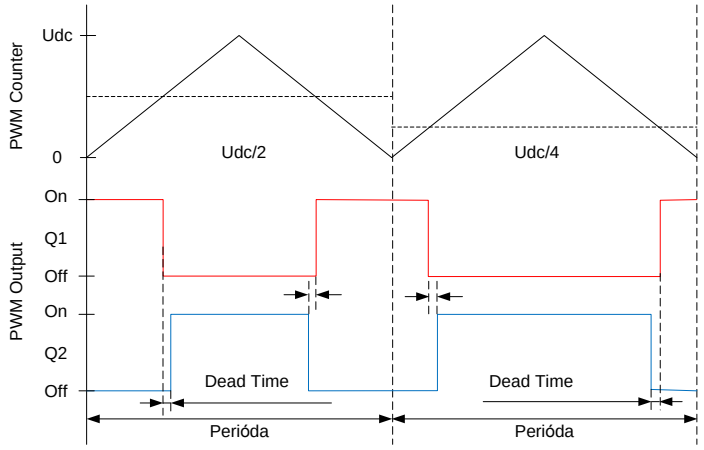
4.3.2 EVM Motor Board

Výkonovú časť riadenia BLDC motora tvorí trojfázový H-mostík zložený zo šiestich spínacích prvkov, Obr. 23. Mostík je napájaný zo zdroju napätia o hodnote 12V, čim sa dá dosiahnuť rozdielové napätie na svorkách motora ±12V. Spínacie členy mostíka tvoria dvojice MOSFET tranzistorov, chránených diódami zapojenými v závernom smere. Jednotlivé dvojice spínacích prvkov sú spúšťané v centrálne zarovnanom komplementárnom móde, v ktorom je vždy jeden z tranzistorov v otvorenom stave, až na výnimku vkladaného Dead Timu.



Obr. 23 Trojfázový H-mostík [7]

Dead Time je časové obdobie vkladané do priestoru prepnutia tranzistorov, poskytujúce čas k úplnému zatvoreniu vypínaného tranzistora pred zopnutím komplementárneho tranzistora. Dead Time zamedzuje vyskratovaniu napájacieho zdroja s následným fyzickým zničením výkonového H-mostíku. EVM motor Board obsahuje spínacie prvky Si4558DY ktorých prepínací čas je 55ns. Použili sme preto nastavenie Dead Timu v PWMMC beane na hodnotu 1µs. Výslednú spínaciu frekvenciu H-mostíka použitú v cieľovej aplikácií sme nastavili na 16 kHz. Ukážka centrálne zarovnaného komplementárneho spínania napätí H-mostíkom je znázornené na Obr. 24.



Obr. 24 Komplementárne spínanie tranzistorov H-mostíka s vloženým Dead Time

4.4 Implementácia riadiaceho algoritmu BLDC motora

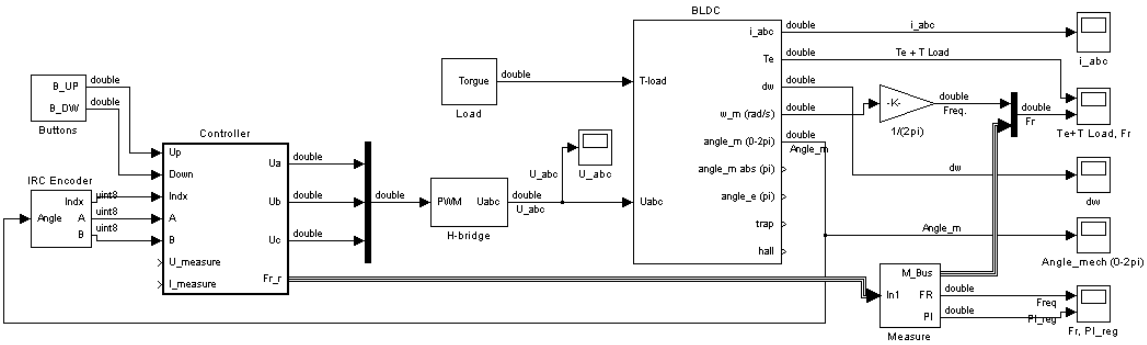
Pri realizácií riadenia BLDC motora boli vytvorené dve demonštračné úlohy. Obidve úlohy realizujú riadenie otáčok rotora BLDC, v uzavretom regulačnom obvode s PI regulátorom. Demonštračné úlohy sa líšia metódami merania frekvencie otáčania rotora. Prvá realizácia riadenia, používa k zisťovaniu aktuálnej frekvencie rotora, hodnotu získanú z IRC senzorov a QD dekodéru. Druhá realizácia riadenia, používa k meraniu frekvencie rotora Capture čítače cieľového procesora a hodnotu výstupu Hall senzorov. V demonštračných úlohách boli použité bloky knižnice Peslib, k simulovaniu vstupných a výstupných periférií mikrokontrolérov. Po overení funkčnosti pomocou simulácie v uzavretom regulačnom obvode, bol model regulátora použitý k vygenerovaniu C kódu, ktorý bol následne preložený a naprogramovaný do MCU. Postup je opísaný v odstavci 2.3.2. Pomocou experimentov na skutočnom motore bola overená funkčnosť regulátora implementovaného do MCU.

K demonštračným úlohám bola vytvorená vizualizácia s možnosťou ovládania cieľovej aplikácie cez zbernicu RS-232 procesorovej dosky MC56F8367EVM. K vytvoreniu vizualizácie a meraniu dát bol použitý program FreeMaster zdokumentovaný v odstavci 2.4

4.4.1 Riadenie BLDC so spätnou väzbou od IRC a QD

Na Obr. 25, je zobrazená simulačná schéma riadenia BLDC motora regulátorom, so spätnou väzbou od IRC senzora a QD dekodéra. Okrem systému BLDC motora so

záťažou a regulátora, sa pri simulácií využíval simuláčný blok IRC senzora, ktorý vytvára výstupné pulzy IRC. Na výslednej aplikácií realizovanej na cieľovom HW sa požadovaná hodnota frekvencie rotora nastavuje tlačením tlačidiel na procesorovej doske, poprípade zadávaním hodnôt žiadanej hodnoty frekvencie cez vizualizáciu vytvorenú v programe FreeMaster . V rámci simulácie sa zadávanie hodnôt frekvencie tlačidlami, simuluje blokom Buttons.



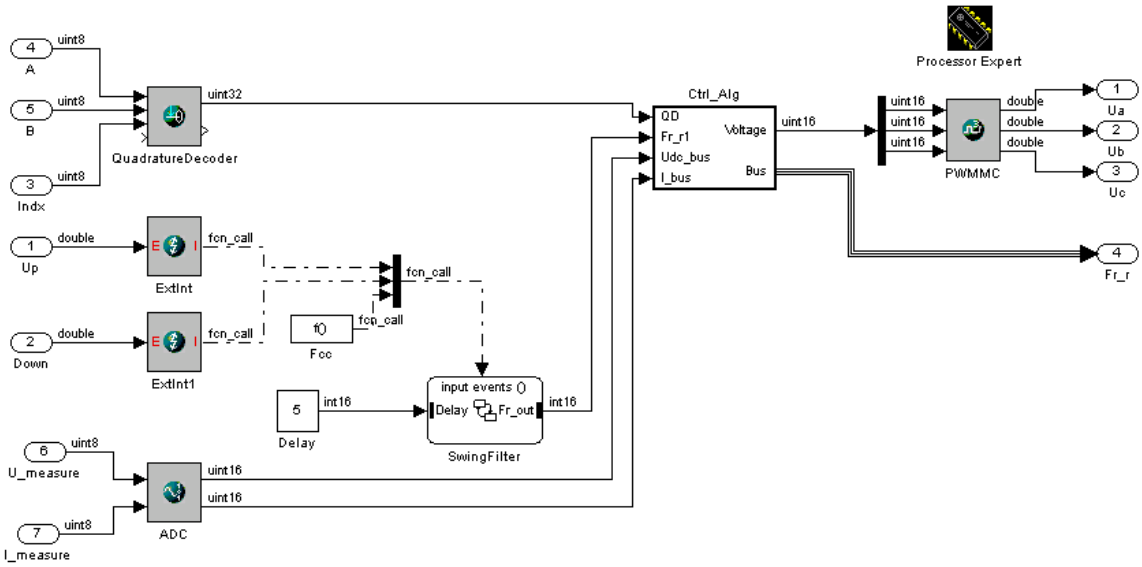
Obr. 25 Simulačná schéma riadenia BLDC so spätnou väzbou od IRC a QD

Na Obr. 26 je vyobrazená schéma regulátora, obsahujúca nasledujúce časti:

- Bloky vstupných a výstupných periférií mikroprocesora knižnice Peslib:
  - blok QuadratureDecoder slúži k získavaniu polohy rotora, z ktorej sa počíta aktuálna frekvencia otáčok.
  - blok ExtInt ošetruje prerušenie procesora od externej udalosti. Vyvolaním prerušenia sa aktivuje blok SwingFilter.
  - blok ADC predstavuje A/D prevodník využívaný k meraniu napätia a prúdu dodávaného napájacím zdrojom do BLDC motora na cieľovom HW. V rámci simulácie sa tento blok nevyužíva.
  - blok PWMMC slúži k simulovaniu chovania trojfázového PWM generátora
- Blok riadiaceho algoritmu regulátora Ctrl_Alg. Blok riadiaceho algoritmu je spúšťaný cyklicky s pevne nastavenou frekvenciou. Takéto spúšťanie riadiaceho algoritmu sa nazýva Time triggered [6]. Podrobnejšie popísanie bloku riadiaceho algoritmu je v nasledujúcom texte.

- Blok SwingFilter: Stavový automat SwingFilter sa používa k počítaniu požadovanej hodnoty frekvencie, nastavovanej prerušením od tlačidiel. Zároveň zabraňuje viacnásobnému napočítaniu hodnoty frekvencie, pri jednom stlačení tlačidla.

Na Obr. 26 vidno vstupné a výstupné dátové typy jednotlivých periférií mikroprocesora, z ktorými sme museli počítať v simuláciách, aj v reálnej aplikácií na cieľovom procesore.



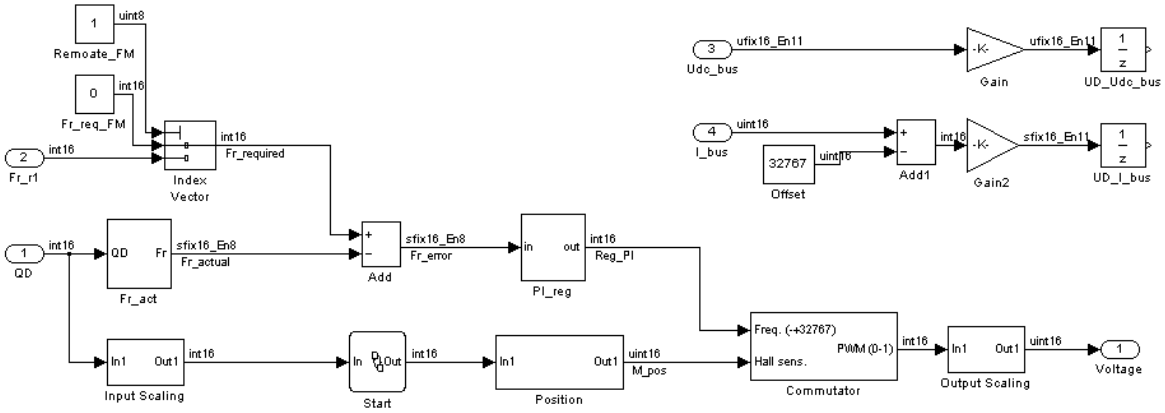
Obr. 26 Simulačná schéma subsystému regulátora so spätnou väzbou od IRC a QD

Vnútorná štruktúra riadiaceho algoritmu Ctrl_Alg je znázornená na Obr. 27. Štruktúra riadiaceho algoritmu, z ktorej bol generovaný kód výslednej aplikácie je zdokumentovaná v Príloha C. Výpočty prevádzané v časti regulátora a riadiaceho algoritmu, sú prevádzané s použitím dátových typoch 16 a 32 bit Integer, poprípade 16 bit Signed Fixed-Point Integer. Týmto krokom sme zabezpečili simuláciu aritmeticko-logickej jednotky cieľového procesora.

Riadiaci algoritmus sa skladá z nasledujúcich častí:

- blok Fr_act: zabezpečuje výpočet aktuálnej frekvencie otáčania rotora.

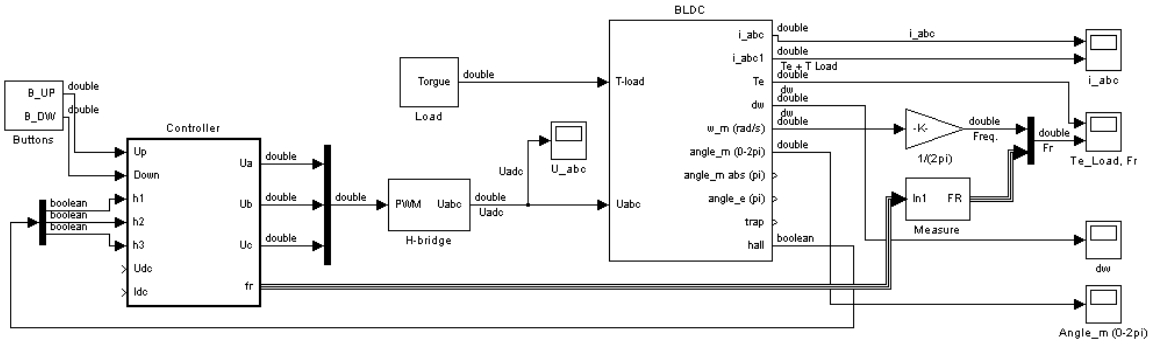
- bloky Imput Scaling, Start, Position: zabezpečujú výpočet aktuálnej polohy rotora spôsobom cyklického porovnávania hodnoty QD dekodéra s hranicami komutácie. Blok Start vykonáva úvodnú inicializáciu polohy rotora pri prvom spustení programu.
- blok PI_reg: obsahuje PI regulátor slúžiaci k regulovaniu frekvencie otáčok rotora.
- blok Commutator: zabezpečuje komutáciu výstupných svorkových napätí BLDC motora. Blok Commutatior obsahuje komutačné tabuľky v ktorých sú vyhľadávané kombinácie výstupných svorkových napätí, záležiacich na aktuálnej polohe rotora. Smer komutácie sa určuje z výstupnej hodnoty PI regulátora.
- Blok Index Vector: slúži ako prepínač zdrojov požadovanej hodnoty regulovanej frekvencie. Určuje, či sa bude regulovať výstupná frekvencia otáčok rotora na hodnotu zadanú tlačidlami, alebo hodnotou nastavovanou skrz vytvorenú vizualizáciu v programe FreeMaster.



Obr. 27 Simulačná schéma riadiaceho algoritmu BLDC so spätnou väzbou od IRC a QD

4.4.2 Riadenie BLDC so spätnou väzbou od Hall senzorov

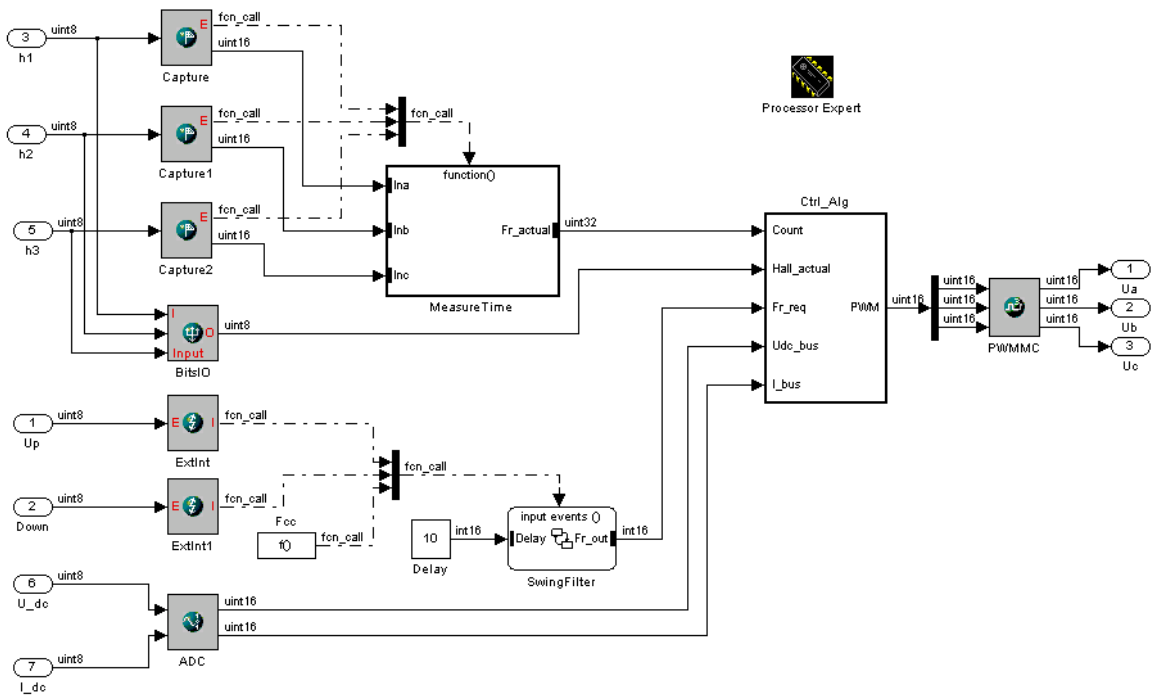
Celková regulačná schéma BLDC motora riadeného regulátorom, so spätnou väzbou od Hall senzorov je znázornená na Obr. 28. Význam jednotlivých blokov použitých v tejto regulačnej schéme je rovnaký ako pri predošlom prípade v odstavci 4.4.1. Jediný rozdiel je v absencií bloku simulujúceho IRC senzor, nahradeného spätnou väzby regulátora hodnotou od Hall senzorov BLDC.



Obr. 28 Simulačná schéma riadenia BLDC so spätnou väzbou od Hall senzorov

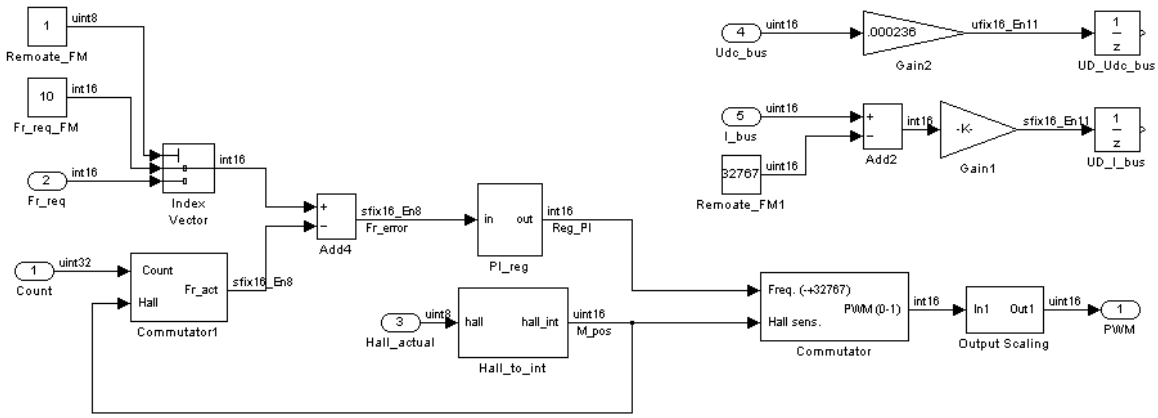
Na Obr. 29 je vyobrazená schéma regulátora, ktorý obsahuje nasledujúce časti:

- Blok riadiaceho algoritmu regulátora Ctrl_Alg, spúšťaný v režime Time triggered.
- Bloky vstupných a výstupných periférií mikroprocesora knižnice Peert:
  - bloky Capture slúžia k vyvolávaniu prerušení pri zmene logickej hodnoty výstupných signálov Hall senzorov. S pomocou blokov Capture bola implementovaná technika meranie frekvencie rotora zdokumentovaná v odstavci 4.2.3.2. Vyvolaním prerušenia sa aktivuje blok.
  - blok ExtInt ošetruje prerušenie procesora od externej udalosti MeasureTime.
  - blok ADC predstavuje A/D prevodník využívaný k meraniu napätia a prúdu dodávaného napájacím zdrojom do BLDC. V rámci simulácie sa tento blok nevyužíva.
  - blok PWMMC slúži k simulovaniu chovania trojfázového PWM generátora.
- Blok MeasureTime počíta čas uplynutý medzi jednotlivými zmenami Hall senzorov zachytených prerušením procesora od blokov Capture. Blok MeasureTime je spúšťaný od udalostí. Takéto spúšťanie algoritmov sa nazýva Event triggered [6].
- Blok StateFlow automatu SwingFilter: Stavový automat SwingFilter sa používa k počítaniu požadovanej hodnoty frekvencie od tlačidiel.



Obr. 29 Simulačná schéma riadenia BLDC so spätnou väzbou od Hall senzora

Vnútorná štruktúra riadiaceho algoritmu Ctrl_Alg je znázornená na Obr. 30. Riadiaci algoritmus je podobný, ako v predošlom prípade z odstavca 4.4.1. Jediný rozdiel je vo výpočte aktuálnej hodnoty frekvencie otáčania rotora, ktorá závisí na hodnotách registrov Capture beanov a hodnote Hallových senzorov. Ku komutácií sa priamo využívajú hodnoty Hall senzorov.



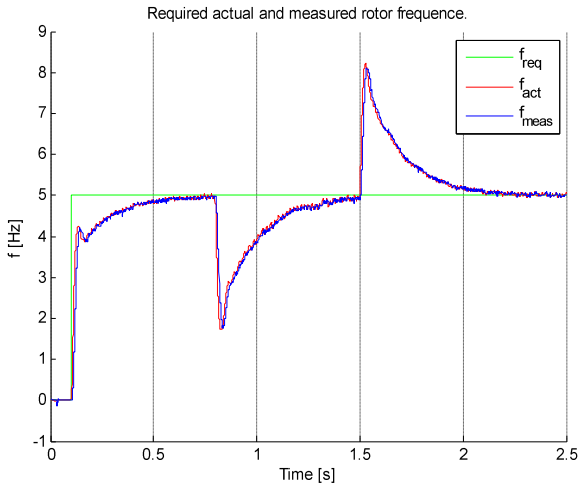
Obr. 30 Simulačná schéma riadiaceho algoritmu BLDC so spätnou väzbou od Hall senzora

5 Vyhodnotenie simulácie a reálneho riadenia BLDC

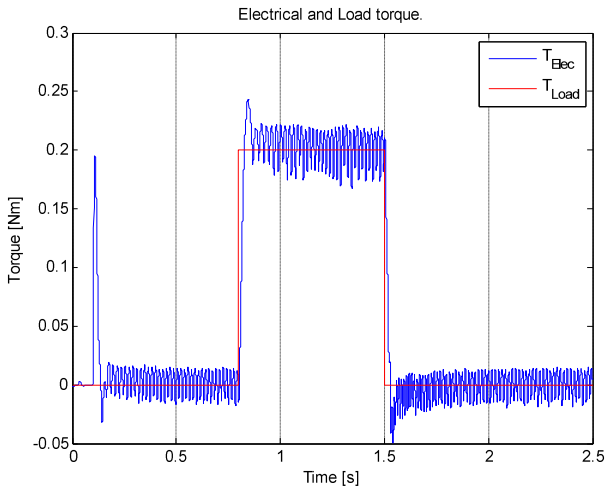
Na modeli BLDC motora sme testovali riadiace algoritmy popísané v odstavci 4.4. Cieľom simulácií bolo ukázať funkčnosť riadiaceho algoritmu na simulačnom modely BLDC. Následne bol z modelu regulátora vygenerovaný C kód pre implementáciu vstavaného regulátoru. Tím bolo možné overiť i funkčnosť automaticky generovaného kódu časti regulátora na cieľovom HW. Výstupné priebehy simulácií riadenia BLDC, sme porovnali s dátami získaných pri meraním na cieľovom HW.

5.1 Simulácia riadenia BLDC so spätnou väzbou od IRC a QD

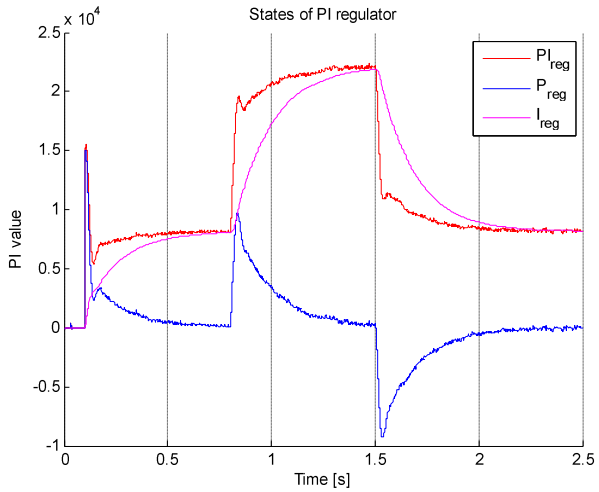
Pri simulácií riadenia BLDC motora, so spätnou väzbou od IRC a QD, sme použili nastavenia PI regulátora s hodnotami proporcionálnej zložky $K_P = 3000$ a integračnej zložky $K_I = 50$. Frekvencia spúšťania bloku riadiaceho algoritmu Ctrl_Alg, bola nastavená na frekvenciu f=500 Hz. Výsledkom prevedených simulácií je odozva regulovanej sústavy, na skokovú zmenu žiadanej hodnoty frekvencie na hodnotu 5Hz, s následným pripojenia záťaže s krútiacim momentom 0.2 Nm. Výstupom simulácii sú priebehy frekvencií otáčania rotora na Obr. 31, záťažové momenty motora na Obr. 32 a stavy akčných zásahov PI regulátora na Obr. 33.



Obr. 31 Simulovaný výstup regulovania frekvencie otáčania rotora, IRC senzor



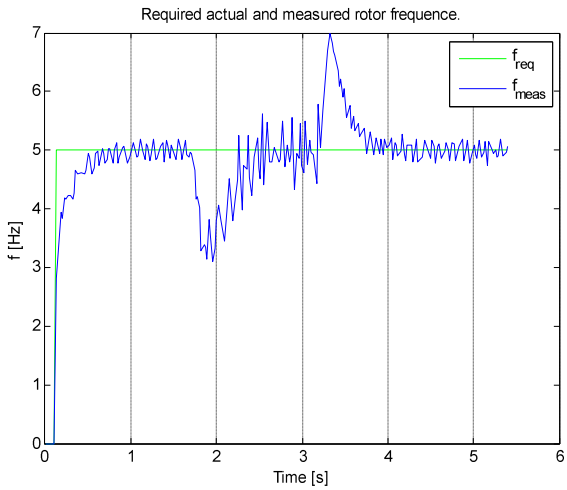
Obr. 32 Simulovaný krútiaci moment záťaže a Ele. krútiaci moment motora, IRC senzor



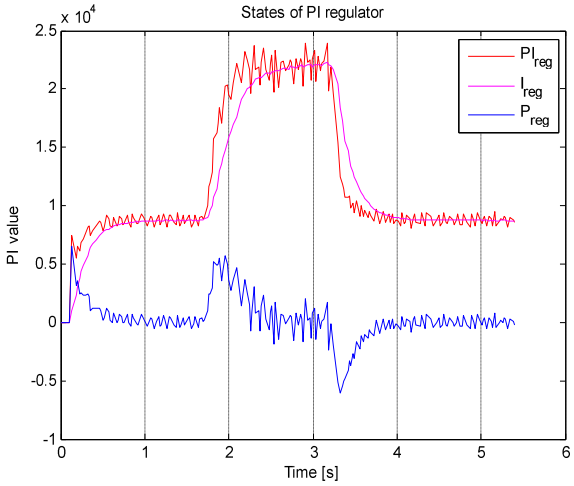
Obr. 33 Simulovaný akčný zásah PI regulátora, IRC senzor

5.2 Reálna regulácia BLDC so spätnou väzbou od IRC a QD

Reguláciu BLDC motora vo výslednej forme vykonávala procesorová doska MC56F8367EVM programovaná automaticky generovaným C kódom, z časti regulátora simulačnej schémy. Výsledky meraní na cieľovom HW, sa získavali s požitím programu FreeMaster. Pri implementácií sme použili rovnaké nastavenia časti riadiaceho algoritmu, ako v simulácií. Výsledkami meraní sú priebehy frekvencie otáčania rotora na Obr. 34 a stavy akčných zásahov PI regulátora na Obr. 35.



Obr. 34 Výsledky merania frekvencie BLDC, IRC senzor



Obr. 35 Výsledky merania akčného zásahu PI regulátora, IRC senzor

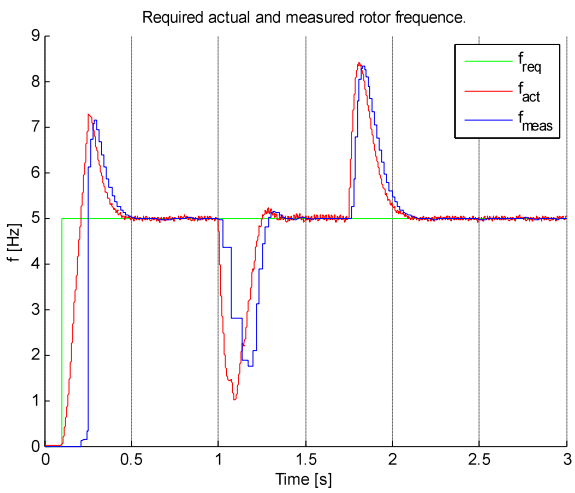
Pri porovnaní výsledkov simulovaných a meraných dát z cieľového HW, vidno podobnosť priebehov regulovanej frekvencie otáčania rotora a akčných zásahov PI regulátora.

Priebehy meraných dát majú väčší rozptyl hodnôt ako pri simulacií. Odchýlky simulovaných a meraných dát by sa dali vysvetliť rozdielmi riadeného motora od simulačného modelu BLDC. Prípadne použitím nevhodnej metódy spracovania hodnôt QD dekodéra. Model BLDC je popísaný ideálnymi matematickými rovnicami, ktoré

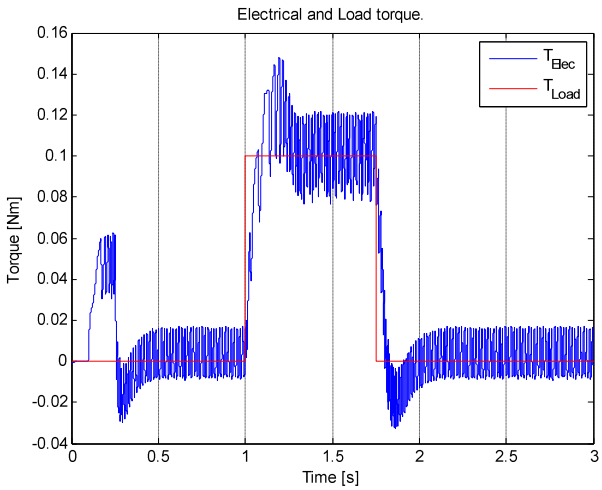
nezahŕňajú rôzne parazitné vplyvy, ako napríklad suché trenie ložísk motora, externé rušivé javy regulácie. Napriek malým nezrovnalostiam, simulácia ukázala funkčnosť riadiaceho algoritmu, ktorý bol spustiteľný a funkčný aj na cieľovom HW.

5.3 Simulácia riadenia BLDC so spätnou väzbou od Hall senzorov

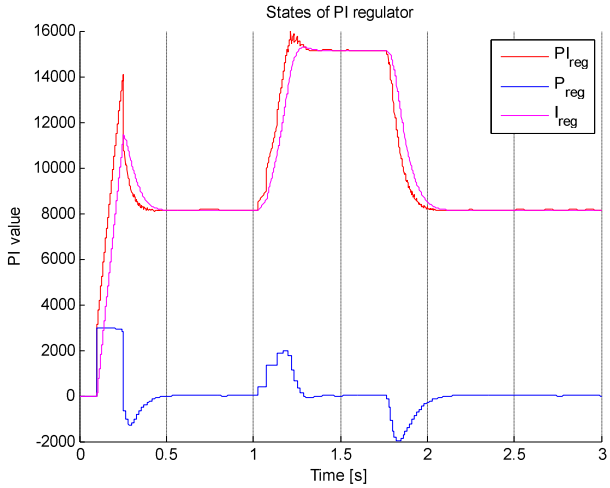
Pri simulácií riadenia BLDC motora, so spätnou väzbou od Hall senzorov, sme použili nastavenia PI regulátora s hodnotami proporcionálnej zložky $K_P = 600$ a integračnej zložky $K_I = 30$. Frekvencia spúšťania bloku riadiaceho algoritmu Ctrl_Alg, bola nastavená na frekvenciu f=500 Hz. Výsledkom prevedených simulácií je odozva regulovanej sústavy, na skokovú zmenu žiadanej hodnoty frekvencie, na hodnotu 5Hz, s následným pripojenia záťaže s krútiacim momentom 0.1 Nm. Výstupom simulácii sú priebehy frekvencií otáčania rotora na Obr. 36, záťažové momenty motora na Obr. 37 a stavy akčných zásahov PI regulátora na Obr. 38.



Obr. 36 Simulovaný výstup regulovania frekvencie otáčania rotora, Hall senzor

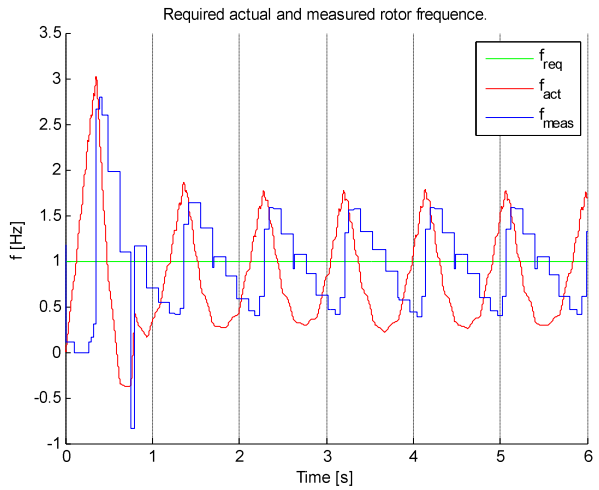


Obr. 37 Simulovaný krútiaci moment záťaže a Ele. krútiaci moment motora, Hall senzor

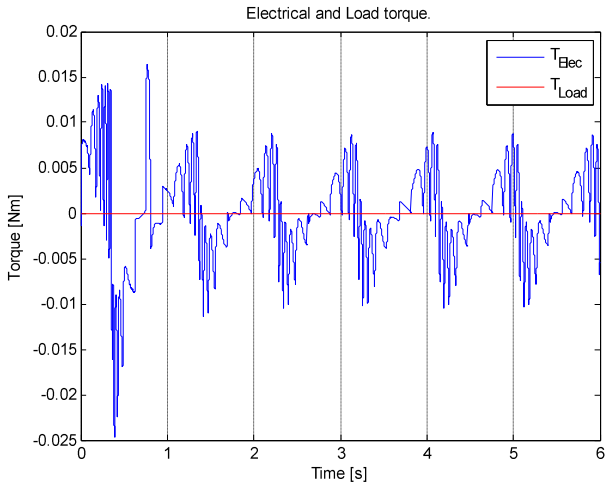


Obr. 38 Simulovaný akčný zásah PI regulátora, Hall senzor

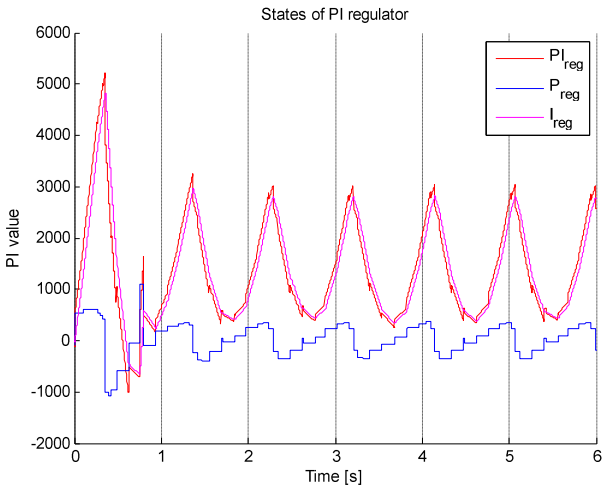
Na rovnakom nastavení regulátora bola prevedená simulácia chovania motora pri skokovej zmene žiadanej hodnoty frekvencie, na hodnotu 1Hz, bez pripojenia záťaže. Výstupom simulácii sú priebehy frekvencií otáčania rotora na Obr. 39, záťažové momenty motora na Obr. 40 a stavy akčných zásahov PI regulátora na Obr. 41.



Obr. 39 Simulovaný výstup regulovania frekvencie otáčania rotora, Hall senzor, f_{req}=1Hz



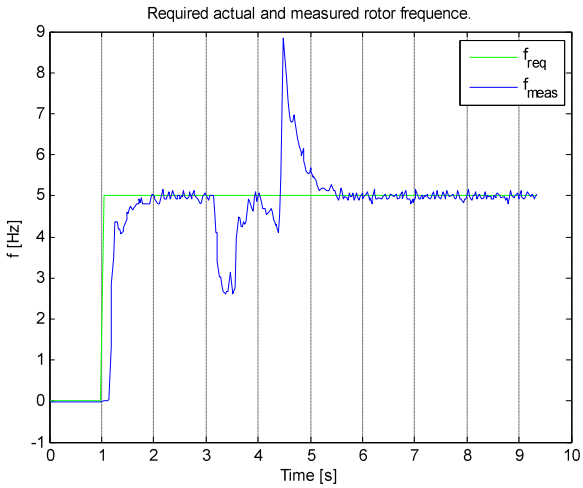
Obr. 40 Simulovaný krútiaci moment záťaže a Ele. krútiaci moment motora, Hall senzor, f_{req}=1Hz



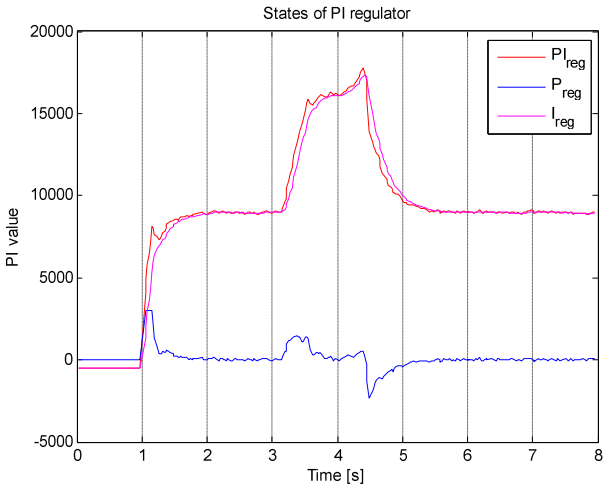
Obr. 41 Simulovaný akčný zásah PI regulátora, Hall senzor, f_{req}=1Hz

5.4 Reálna regulácia BLDC so spätnou väzbou od Hall senzorov

Programovanie cieľovej aplikácie a meranie priebehov regulácie BLDC bolo prevedené rovnakým spôsobom ako v predošlom prípade. Pri implementácií sme použili rovnaké nastavenia časti riadiaceho algoritmu, ako v simulácií. Výsledkami meraní sú priebehy frekvencie otáčania rotora na Obr. 42 a stavy akčných zásahov PI regulátora na Obr. 43.

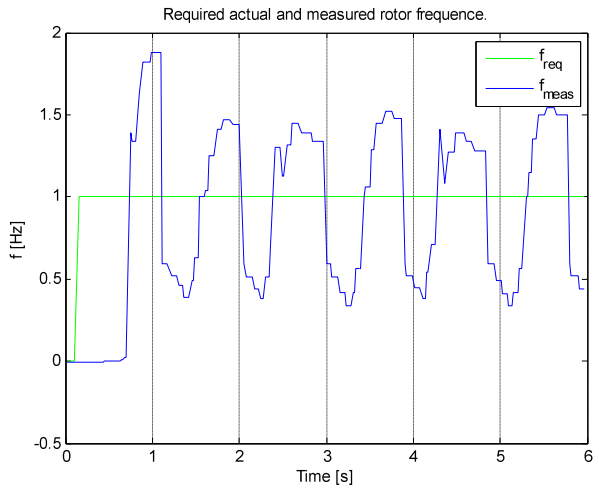


Obr. 42 Výsledky merania frekvencie BLDC, Hall senzor

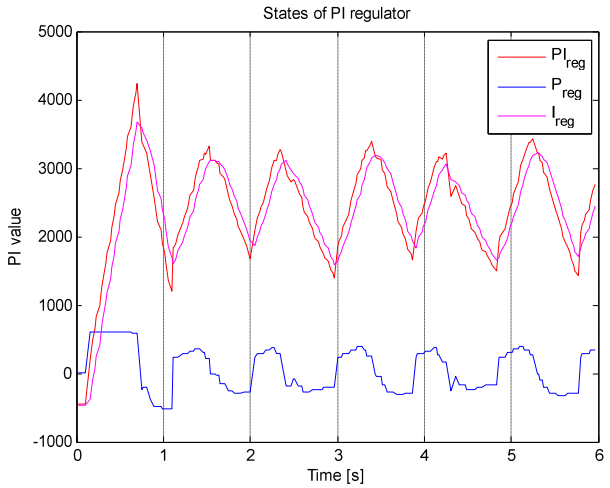


Obr. 43 Výsledky merania akčného zásahu PI regulátora, Hall senzor

Pre skokovú zmenu žiadanej frekvencie na hodnotu 1Hz sme získali meraním priebehy frekvencie otáčania rotora na Obr. 44 a stavy akčných zásahov PI regulátora na Obr. 45.



Obr. 44Výsledky merania frekvencie BLDC, Hall senzor, f_{req}=1Hz



Obr. 45 Výsledky merania akčného zásahu PI regulátora, Hall senzor, f_{req}=1Hz

Tak ako v odstavci 5.2, aj v tomto prípade simulácia preukázala funkčnosť riadiaceho algoritmu, ktorá bola spustiteľná a funkčná na cieľovom HW. Na Obr. 36 vidno zo simulovaného priebehu aktuálnej a meranej frekvencie rotora, časové omeškanie merania frekvencie, ktoré je spôsobené použitím priemerovania hodnoty Capture registrov, použitých k meraniu frekvencie rotora. Nepriaznivo sa v meraní frekvencie rotora prejavil fakt, že hodnota aktuálnej frekvencie rotora, meraná Capture registrami, sa obnovovala len pri zmene hodnoty Hall senzorov, poprípade pri pretečení registrov na pozadí bežiacich

čítačov beanov Capture. Pri nízkych frekvenciách $f < 2Hz$ spôsobovala znížená vzorkovacia frekvencia nestabilituriadenej frekvencie otáčania motora, ktorá bola preukázaná v simulácií (Obr. 39) a v reálnom meraní (Obr. 45).

6 Záver

Účelom tejto diplomovej práce bolo oboznámenie sa s technológiou automatického generovania kódu z prostredia Simulink a prostredia Processor Expert, s následným aplikovaním týchto znalostí, pri rozšírení podpory HW periférií knižnice Peslib. Na demonštračnej úlohe sme mali ukázať funkčnosť návrhu riadiacich algoritmov s použitím knižnice Peslib a overiť použiteľnosť automaticky generovaného kódu z časti regulátora na cieľovej aplikácií.

K úplnosti podpory základných HW periférií mikroprocesorov, chýbal v knižnici Peslib simulačný blok PWMMC. Knižnicu sme z využitím S-Funkcii rozšírili o simulačný blok hardware periférie mikroprocesora PWMMC, ktorá simulovala jej chovanie v základných parametroch. Pre PWMMC blok sme vytvorili podporu automatického generovania C kódu zo Simulinku, ktorého funkčnosť sme otestovali na demonštračnej úlohe riadenia BLDC motora.

V rámci aplikácie riadiacich algoritmov BLDC motora, sme vytvorili dve plne funkčné demonštračné úlohy. Realizácie úloh sa líšili v použitých blokoch knižnice Peslib a v spôsoboch merania aktuálnej frekvencie otáčania rotora. K meraniu frekvencií otáčania rotora, sme použili časom a udalosťami spúšťané funkčné bloky. Po úspešnom otestovaní úloh v Simulinku, sme zo subsystému regulátora vygenerovali riadiaci program nahrávaný do cieľového hardware. Meraniami na demonštračných úlohách sme ukázali správnosť simulácií.

K demonštračným úlohám bola vytvorená vizualizácia regulačného procesu v programe FreeMaster, s možnosťou nastavovania a real-time sledovania jednotlivých parametrov regulácie. Nastavovaním parametrov PI regulátora cez ovládací panel vizualizácie, sme urýchlili proces ladenia výsledného produktu.

V PWMMC bloku sme nemohli implementovať techniku maskovania jednotlivých výstupov PWM a to z dôvodu absencie podpory volania metód PWMMC, blokom Method knižnice Peslib. PWMMC blok by bolo vhodné rozšíriť o techniku maskovania výstupných portov, ktorá sa používa pri riadení BLDC motorov a uľahčuje proces komutácie statorových napätí motora. Ďalší rozvoj knižnice Peslib, by bolo vhodné v budúcnosti smerovať k spresňovaniu simulačných parametrov jednotlivých periférií, s rozširovaním

podpory volania metód v prostredí Simulinku. Ďalej by bolo vhodné vytvoriť podporu komunikačných periférii mikrokontrolérov v knižnici Peslib, pre zbernice CAN a RS232.

7 Slovník použitých termínov a skratiek

A/D prevodník	Analogovo/Digitálny prevodník
BLDC motor	Brushless DC motor
CAN	Controller Area Network
CW	CodeWarrior
DC motor	Direct Current motor
EB	Embedded Bean
EM pole	ElektroMagnetické pole
ERT	Embedded Real-Time Workshop Target
GUI	Graphical User Interface
HAL	Hardware Abstract Layer
HS	Hall Senzor
HW	Hardware
I/O	Input/Output
I2C	Inter-Integrated Circuit
IRC	Incremental Radial Counter
MAC	Multiplier Accumulator
MCU	Microprocessor Control Unit
OOP	Object-Oriented Programming
PC	Personal Computer
PE	Processor Expert
PEERT	Processor Expert embedded Real-Time Target
PI regulátor	Proporcionálno-Integračný regulátor
PMSM	Permanent Magnet Synchronous Motor
PWM	Pulse-Width Modulation
QD	Quadrature Decoder
RAD	Rapid Application Design
RTW	Real-Time Workshop
RTW EC	Real-Time Workshop Embedded Coder
SCI	Serial Communication Interface
SPI	Serial Peripheral Interface
TLC	Target Language Compiler

8 Použitá literatúra a zdroje:

[1] Simulink 7 - Writing S-Functions. The MathWorks, Inc., www.mathworks.com, 2007

[2] Real-Time Workshop - User's Guide. The MathWorks, Inc., www.mathworks.com, 2004

[3] Real-Time Workshop Embedded Coder Developing Embedded Targets. The MathWorks, Inc., www.mathworks.com, 2007

[4] Stateflow and Stateflow Coder User's Guide. The MathWorks, Inc., www.mathworks.com, 2007

[5] Bartosinski R., Stružka P. ,Waszniowski L. (2005): PEERT- Blokset for Processor Expert and Matlab/Simulink integration.

[6] Bartosinski R., Hanzálek Z, Stružka P., Waszniowski L. (2007): Integrated Environment for Embedded Control Systems Design

[7] 3-Phase BLDC Motor Control with Quadrature Encoder using 56F800/E. AN1961, Rev.3, www.freescale.com, 2005

[8] 56F8367/56F8167 Data Sheet. Rev.8, Freescale Semiconductors, www.freescale.com, 2007

[9] Processor Expert help. UNIS, spol. s r.o., www.processorexpert.com, 2008

[10] CodeWarrior. Freescale Semiconductors, www.freescale.com, 2007

[11] HEDS-564X Data Sheet. Agilent Technologies, www agilent.com, 2002

[12] 56F8367 Evaluation Module User Manual. Freescale Semiconductors, www.freescale.com, 2005

[13] 56F8346 Hybrid Controller. 8346BLDCQETD/D, Rev.1, Freescale Semiconductors, www.freescale.com, 2003

[14] Evaluation Motor Board - User's Manual.MEMCEVBUM/D Freescale Semiconductors, www.freescale.com, 2000

[15] Motion Control Group. www.motioncontrolgroup.com

[16] Pavelka J., Čeřovský Z., Javůrek J.: Elektrické pohony, Vydavatelství ČVUT, 2001

[17] FreeMaster for Embedded Applications-User guide. FMSTERUG, Rev.2, Freescale Semiconductors, www.freescale.com, 2007

[18] 3-Phase PM Synchronous Motor Vector Control Using a 56F80x, 56F8100, or 56F8300 Device, AN1931, Freescale Semiconductors, www.freescale.com, 2007

[19] The MathWorks, Inc., www.mathworks.com

[20] www.wikipedia.com

[21] www.wikipedia.cz

Príloha A. Matematický popis a simulačná schéma PMSM

Rovnica 9 Matematický popis elektrickej časti PMSM motora v d-q súradných sústavách [19], [16], [18]

$$\frac{d}{dt}i_d = \frac{1}{L_d}(v_d - Ri_d + L_q p\omega_r i_q)$$
$$\frac{d}{dt}i_q = \frac{1}{L_q}(v_q - Ri_q + L_d p\omega_r i_d - \lambda p\omega_r)$$
$$T_e = 1.5p(\lambda i_q + (L_d - L_q)i_d i_q)$$

Kde:

i_q, i_d	prúdy tečúce jednotlivými cievkami statorového vinutia
v_d, v_g	rozdielové napätia cievok statorového vinutia
R	odpor cievky statorového vinutia
L_d, L_q	indukčnosť cievok statorových vinutí v smere d-q
λ	magnetický toku permanentného magnetu
ω_r	uhľová rýchlosť rotora
p	počet pólových párov vinutí motora
T_e	elektromagnetický krútiaci moment

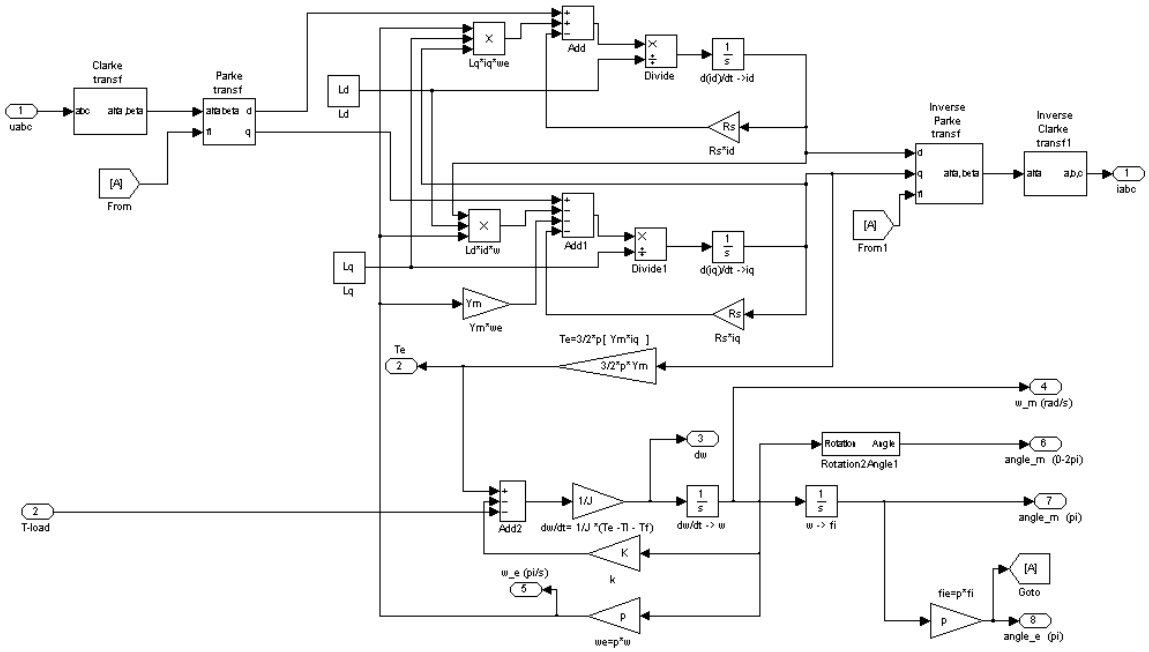
Rovnica 10 Matematický popis mechanickej časti PMSM motora

$$\frac{d}{dt}\omega_r = \frac{1}{J}(T_e - F\omega_r - T_m)$$
$$\frac{d\theta}{dt} = \omega_r$$

Kde :

ω_r	uhlová rýchlosť rotora
θ	uhol natočenia rotora
J	kombinovaná zotrvačnosť statora a záťaže

p	počet pólových párov vinutí motora
T_e	elektromagnetický krútiaci moment
T_m	mechanický krútiaci moment záťaže
F	viskózne trenie

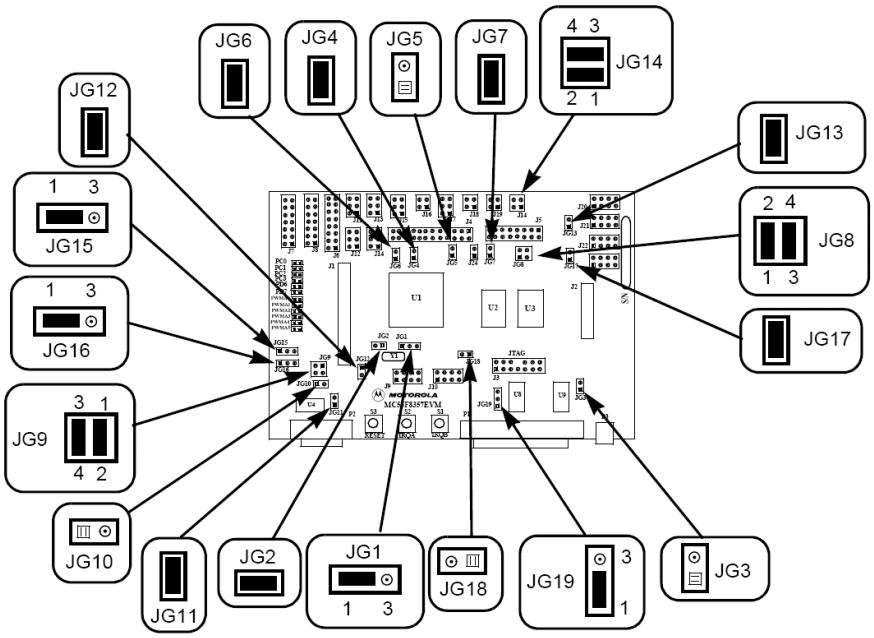


Obr. 46 Simulačná schéma PMSM motora

Príloha B. Popis S-funkcie PWMMC

```
/* Function: mdlOutputs ======================================================
 * Abstract:
 *    PWM + function call
 */
static void mdlOutputs(SimStruct *S, int_T tid)        //metoda mdlOutputs
{
   int_T      i;                                  //lokalne premenne
   int_T      k;
   uint16_T    level_16;
   real_T      level;

   PWMMC_user_data_t *ud = ssGetUserData(S);

   if(ud->OnReload==0){k=0;}
   else{k=1;}

   if(ud->RldCntr==ud->Rld-1){                    //nastala udalost Reload
      ud->RldCntr =0;                             //vynulujeme Reload citac

      for (i=0; i<NINPUTS; i++) {                 //pre vsetky vstupy prevedieme nastavenie vystupov
         InputPtrsType       u       = ssGetInputPortSignalPtrs(S,i);          //ziskanie pointra na vstup
         real_T              *outPt  = ssGetOutputPortRealSignal(S,i+k);     //ziskanie pristupu na vystup

         InputInt16PtrsType   uPt =(InputUInt16PtrsType) u;

         switch(ud->resolution){      //podla rozlisenia PWMMC generatora prevedieme vypocet vystupu
            case 12:                              //rozlisovacia schopnost 12 bitov
               level_16 = *uPt[0];                //ziskanie hodnoty vstupu
               level_16 = level_16 & 0x0fff;   //vymaskovanie hodnoty maximalnym rozlisenim
               level =(real_T) level_16 ;                //prevod hodnoty do datoveho typu real
               outPt[0] = level/0x0fff;                  //podelenie hodnotou maximalneho rozlisenia
               break;
            case 15:                                  //rozlisovacia schopnost 15 bitov
               level_16 = *uPt[0];
               level_16 = level_16 & 0x7fff;
               level =(real_T) level_16 ;
               outPt[0] = level/0x7fff;
               break;
            default:                                  //rozlisovacia schopnost 16 bitov
               level_16 = *uPt[0;
               level =(real_T) level_16 ;
               outPt[0] = level/0xffff;
               break;
         }        }
      if(ud->OnReload==1){                            //ak je aktivovany function-call OnReload
         if (!ssCallSystemWithTid(S,0,tid)) {//vyvolame spustenie subsystemu function-callom
            /* Error occurred which will be reported by Simulink */
            ssSetErrorStatus( S, "PWMMC Function Call error on port 1."); //pri chybe varovanie
            return;
         }       }    }
   else{                                          //pri nenastavenej udalosti Reload
      ud->RldCntr++;                              //inkrementujeme  vnutorny citac
   }}
```

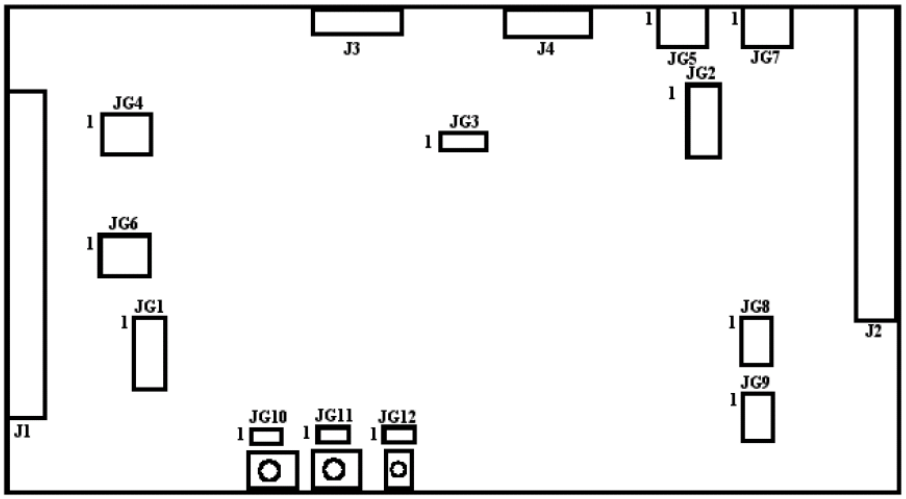
Príloha C. Nastavenie konektorov procesorovej dosky MC56F8367EVM



Obr. 47 Rozloženie konektorov vývojovej dosky MC56F8367EVM

Tab. 3 Nastavenie konektorov vývojovej dosky MC56F8367EVM

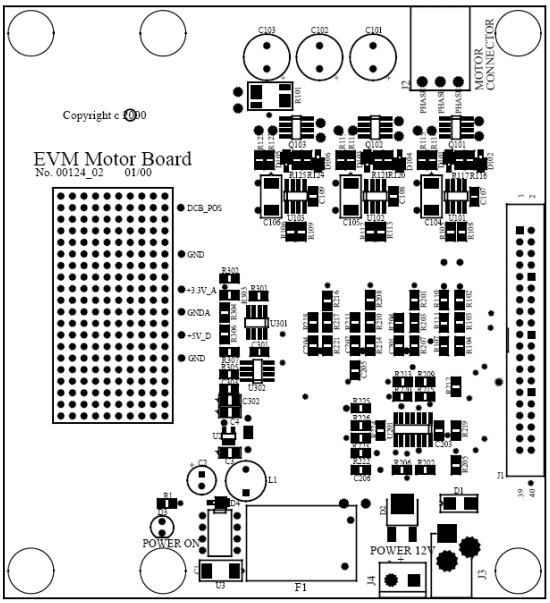
Jumper Groupe	Comment	Conn.
JG1	Use on-board EXTAL crystal for CPU oscillator	1-2
JG2	Use on-board XTAL crystal for CPU oscillator	1-2
JG3	Enable Internal Boot Mode	1-2
JG4	Enable A0-A19 for external memory accesses	NC
JG5	Enable SRAM Memory Bank 0 (use CS0)	1-2
JG6	Enable SRAM Memory Bank 0 (use CS1 & CS2)	
JG7	Enable RS-232 Disable	NC
JG8	SPI #0 Daisy Chain (Optional--not populated on board by default)	NC
JG9	Enable on-board Parallel JTAG Host/Target Interface	NC
JG10	Connect Analog Ground to Digital Ground	NC
JG11	CAN termination selected	1-2
JG12	Pass RXD0 & TXD0 to RS-232 level converter	1-2, 3-4
JG13	Enable Crystal Mode	1-2
JG14	Pass Temperature Diode to ANA7	1-2
JG15	User Jumper #0	1-2
JG16	User Jumper #1	1-2



Obr. 48 Rozloženie konektorov Legacy Motor Interface Daughter Card (LMIDC)

Tab. 4 Nastavenie konektorov Legacy Motor Interface Daughter Card

Jumper Group	Comment	Connections
JG1	Primary PFC	1-2, 3-4, 5-6, 7-8, 9-10
JG2	Secondary PFC	NC
JG3	Phase_IS / Over_I	1-2
JG4	Primary Zero-Crossing / Encoder	2-3, 5-6, 8-9
JG5	Secondary Zero-Crossing / Encoder	2-3, 5-6, 8-9
JG6	Primary Back-EMF / Phase-IS	1-2, 4-5, 7-8
JG7	Secondary Back-EMF / Phase-IS	1-2, 4-5, 7-8
JG8	Fault A Monitor 1-2	1-2, 3-4, 5-6
JG9	Fault B Monitor	1-2, 3-4, 5-6
JG10	Switch 1	1-2
JG11	Switch 2	1-2
JG12	Switch 3 (Run / Stop)	1-2
J1	Primary UNI-3	-
J2	Secondary UNI-3	-
J3	Primary Encoder/Hall-Effect	-
J4	Secondary Encoder/Hall-Effect	-



Obr. 49 Výkonová časť EVM Motor Board (EVM)

Fyzické nastavenie dem. úlohy riadenia BLDC motora so spätnou väzbou od IRC a QD:

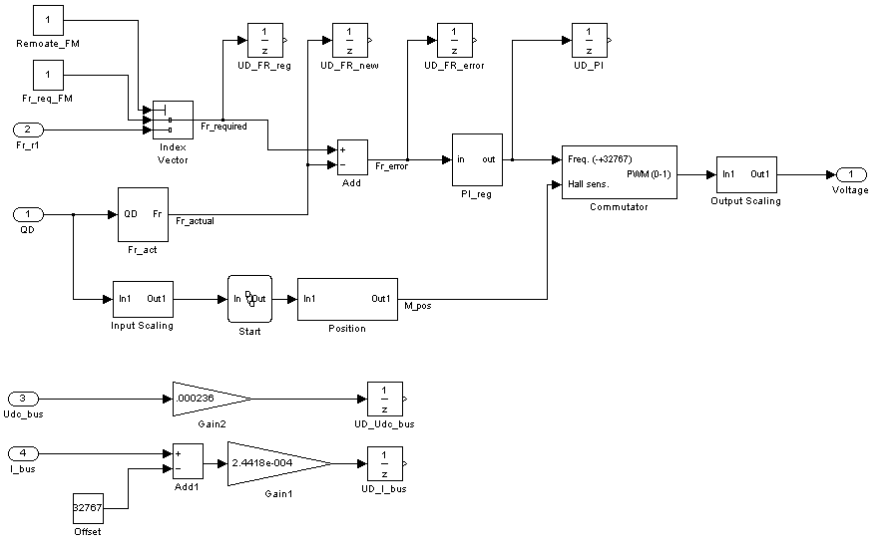
1. Konektor IRC senzora pripojíme na konektor J3 dosky LMIDC
2. Napájací kábel BLDC motora pripojíme na konektor J2 dosky EVM

Fyzické nastavenie dem. úlohy riadenia BLDC motora so spätnou väzbou od Hall senz.:

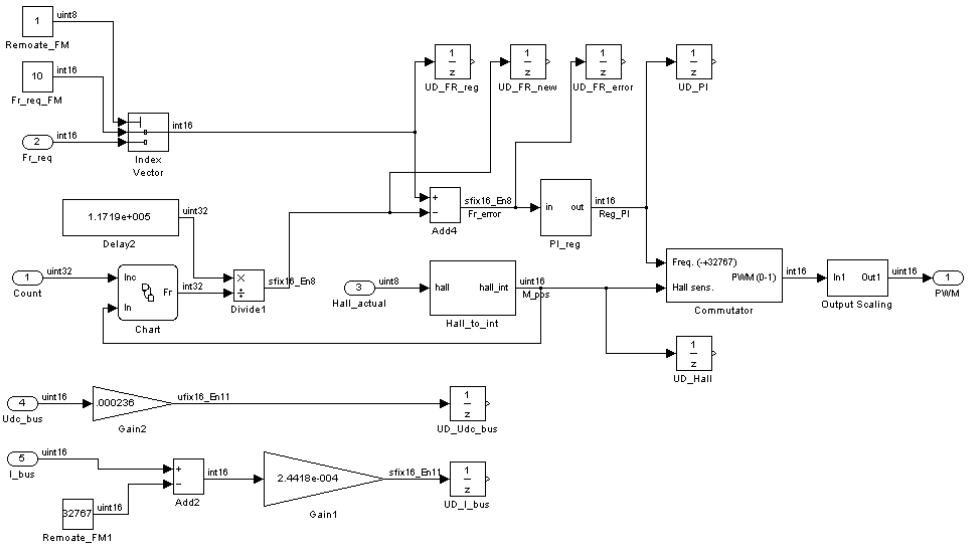
1. Konektor Hall senzora pripojíme na konektor J3 dosky LMIDC
2. Spravíme paralelné prepojenie konektorov J3 a J4 k umožneniu merania frekvencie motora Capture čítačmi cez konektor J4.
3. Napájací kábel BLDC motora pripojíme na konektor J2 dosky EVM

Príloha D. Štruktúra regulátora

K štruktúre riadiaceho algoritmu používanej v simuláciách, boli pridané bloky Unit Delay, simulinkovej knižnice Discrete. Tieto bloky sa po vypnutí optimalizačný nastavení RTW popísaných v odstavci 2.1.2.3 zahŕňali do procesu autamotického generovania C kódu, a sú alokované v rámci dátovej pamäte vo výslednej aplikácií. Dané bloky sa používali ako prístupové body vizualizácie vytvorenej v programe FreeMaster.

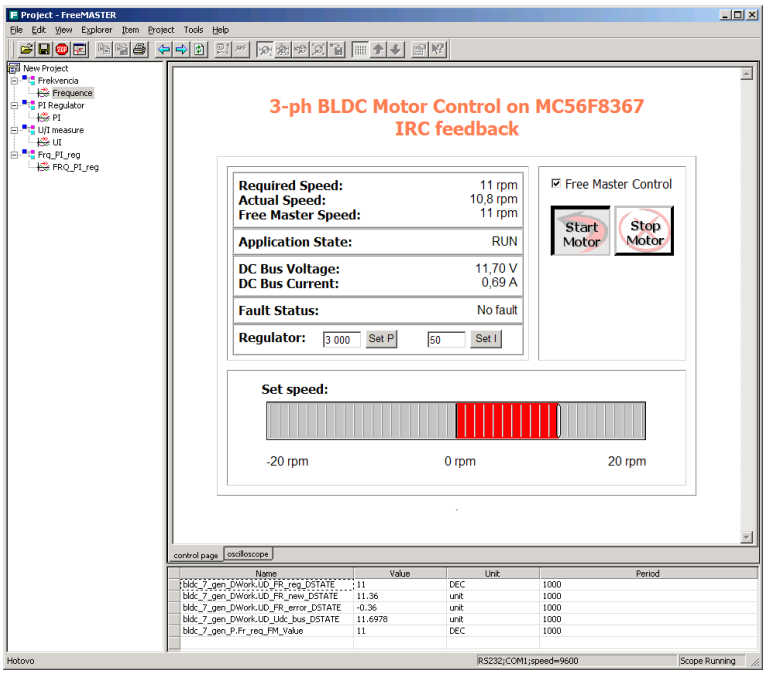


Obr. 50 Štruktúra Ctrl_Alg so spätnou väzbou od IRC a QD

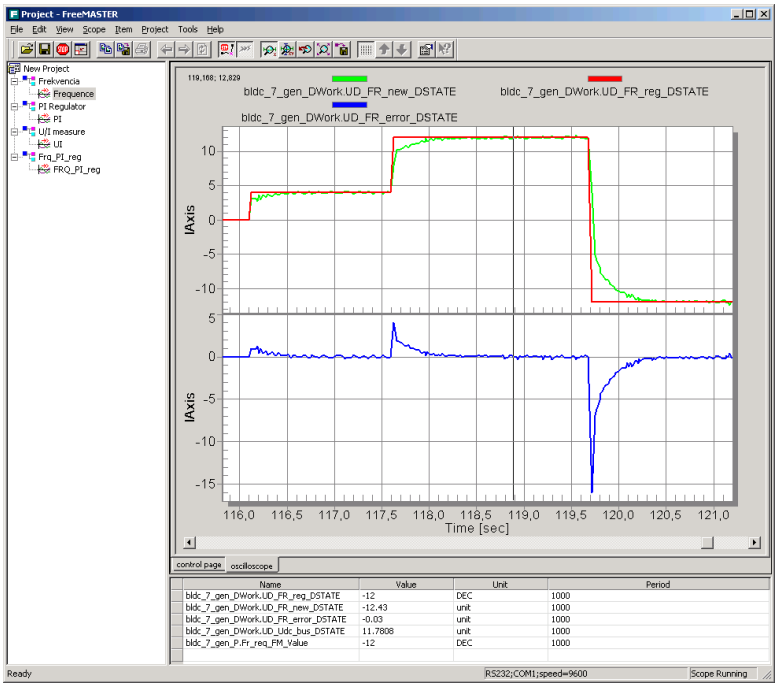


Obr. 51 Štruktúra Ctrl_Alg so spätnou väzbou od Hall senzorov

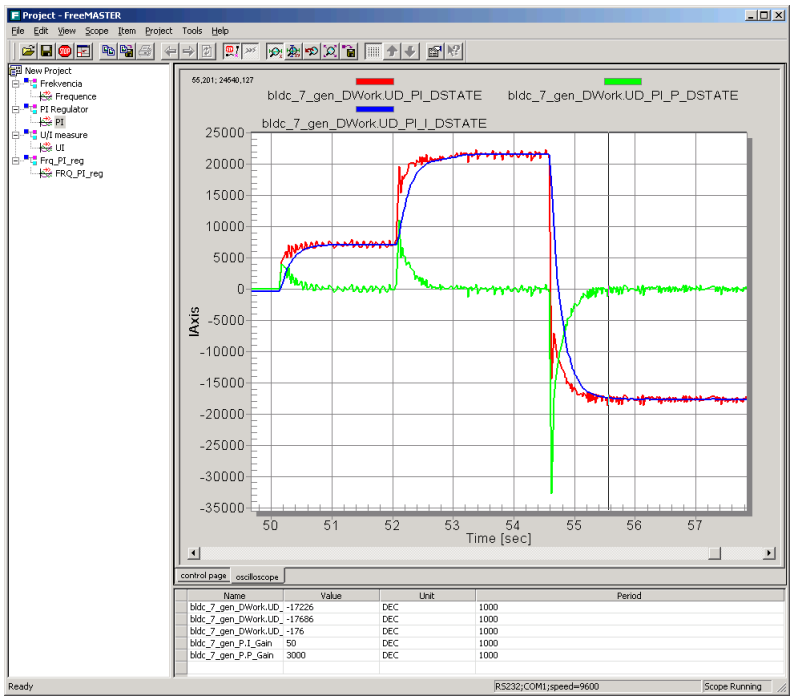
Príloha E. Vizualizácia procesu riadenia BLDC v programe FreeMaster



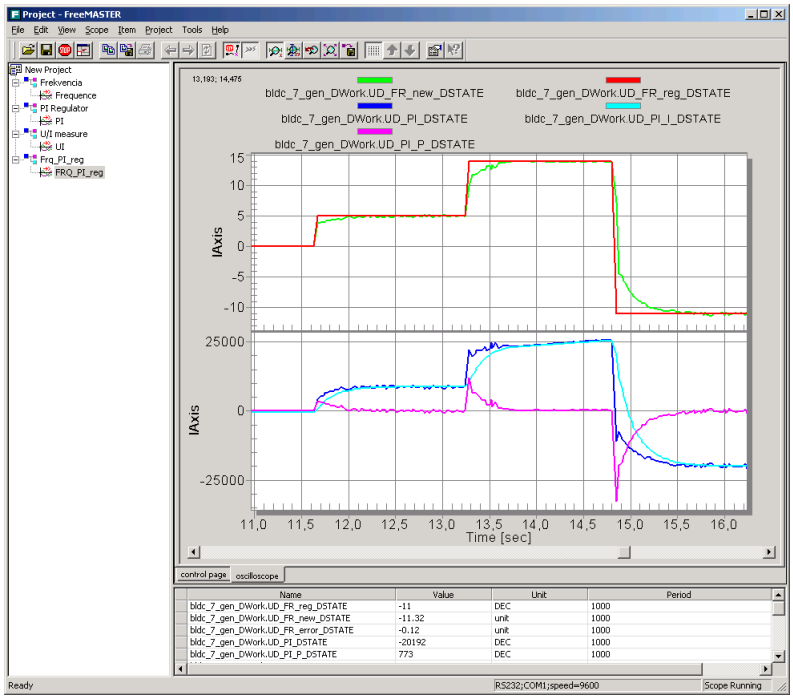
Obr. 52 Ovládací panel vizualizácie riadenia BLDC motora



Obr. 53 Meranie frekvencie otáčok rotora

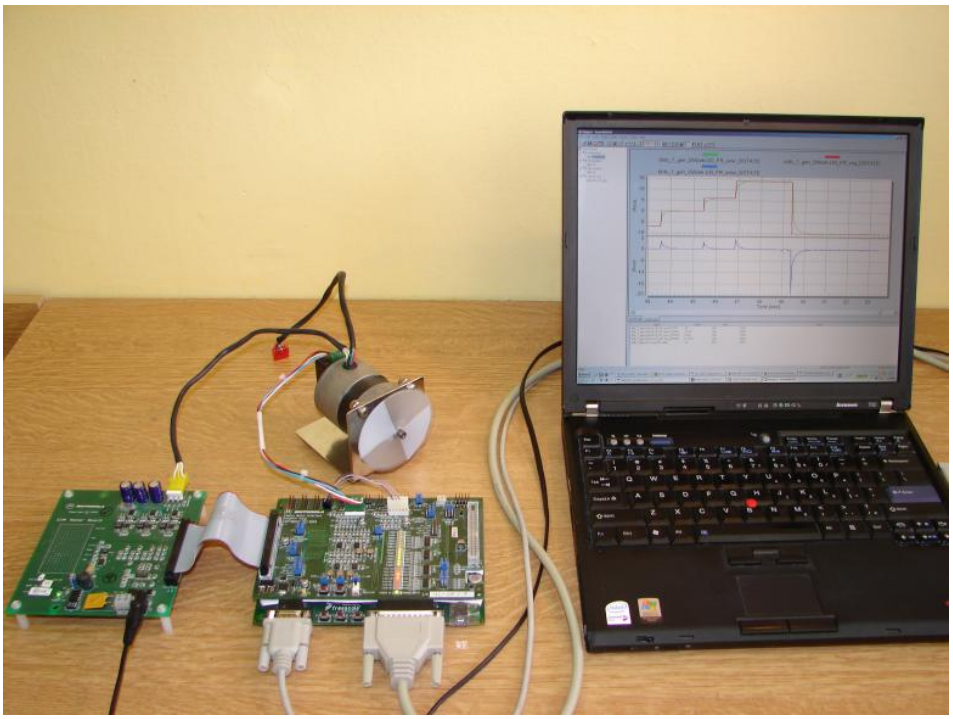


Obr. 54 Meranie akčných zásahov PI regulátora



Obr. 55 Združené meranie frekvencie otáčok rotora a akčných zásahov PI regulátora

Príloha F. Fotodokumentácia



Obr. 56 BLDC motor, riadiaci systém, vizualizácia procesu