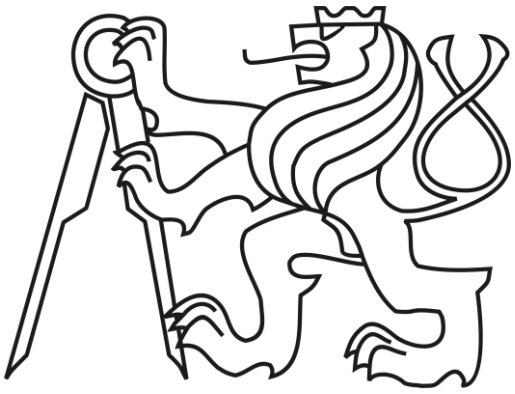
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ

Řízení krokového motoru pomocí systému CompactRIO systémem LabView

Bakalářská práce



Vedoucí práce: Ing. Vít Hlinovský, CSc.
Student: Evžen Konstantinov

Květen 2014

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra elektrických pohonů a trakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Evžen Konstantinov**

Studijní program: Elektrotechnika, energetika a management
Obor: Aplikovaná elektrotechnika

Název tématu: **Řízení krokového motoru pomocí systému CompactRIO systémem LabView**

Pokyny pro vypracování:

1. Instalace systému CompactRIO

2. Sestavení algoritmu pro řízení elektrického pohonu

3. Vytvoření uživatelského prostředí v LabView

Seznam odborné literatury:

[1] Motor control edition – Freescale - Issue 8. 2012 Freescale Semiconductor, Inc.
[2] Manuál LabView

Vedoucí: Ing. Vít Hlinovský, CSc.

Platnost zadání: do konce letního semestru 2014/2015

prof. Ing. Jiří Lettl, CSc.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 3. 12. 2013

Anotace

Tato práce se zaměřuje na problematiku řízení krokových motorů pomocí zařízení CompactRIO a systému LabView od firmy National Instruments. Úvodní část práce popisuje krokové motory, použité řídící elektronické prvky a systém LabView, v němž je realizován řídící algoritmus. Praktická část práce je věnována zapojení a zprovoznění krokového pohonu se zařízením CompactRIO včetně popisu řídícího algoritmu, vytvořeného za pomoci grafického programování v programu LabView. Poslední část práce je soustředěna na vytvoření grafického rozhraní řídícího programu.

Klíčová slova: Krokový motor, LabView, CompactRIO, FPGA

Abstract

This thesis is concentrated on stepper motor control in conjunction with the systems CompactRIO and LabView designed by the company National Instruments. The beginning of the thesis describes stepper motors, control electronics and the system LabView, which is used for creation of the control algorithm. The practical part of this thesis is dedicated to the connection of the stepper drive to the CompactRIO system, as well as description of the control algorithm created using Graphical programming in LabView. At the end of the thesis, the graphical user interface of the control program is described.

Keywords: Stepper motor, LabView, CompactRIO, FPGA

Čestné prohlášení

Prohlašuji, že jsem zadanou bakalářskou práci „Řízení krokového motoru pomocí systému CompactRIO systémem LabView“ vypracoval samostatně a použil jsem k tomu pouze literaturu, kterou uvádím v seznamu přiloženém k této bakalářské práci.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne …….. ……

Evžen Konstantinov

Poděkování

Chtěl bych poděkovat svému vedoucímu bakalářské práce, panu Ing. Vítu Hlinovskému, CSc., za odborné vedení, za pomoc a rady při zpracování této práce. Také děkuji své rodině za podporu při studiu a při vypracování této práce.

Obsah

1. Úvod

Co by si dnešní civilizace počala bez elektřiny? V minulých stoletích vědci dospěli k závěru, že elektrická energie bude nejvhodnější formou energie jak pro výrobu, tak pro přenos. Spolu s objevováním elektromagnetických zákonů byly konstruovány první elektrické stroje. Tyto stroje neoplývaly úžasnými výkonovými parametry, avšak ukázaly lidstvu směr, jímž se lze ve výzkumu ubírat. Když se lidstvo naučilo efektivně vyrobit elektrickou energii, nastal rozmach ve vývoji elektrotechniky, stroje se začaly zdokonalovat. Po uplynutí dekád jsme dospěli do stádia, kdy jsou elektrické stroje a pohony takřka nenahraditelné a mají uplatnění téměř ve všech oborech lidské činnosti.

Elektrické pohony se vyvíjely od nejjednodušších forem (elektromotor připojený ke zdroji elektrického proudu) až po velmi složité soustavy elektrotechnických prvků a přístrojů (elektromotor napájený z polovodičového měniče, zapojený se zpětnou vazbou, se senzory, ochrannými prvky, měřicími přístroji, řídícími počítači atp.). Dnes lze navrhnout pohon přesně podle požadavků a náročnosti aplikace. Existuje mnoho typů motorů, které lze v pohonu využít, každý má své klady i zápory, některé motory lze využít jen v určitých aplikacích.

Zatímco princip funkce elektromotoru je neměnný, velmi variabilní je způsob řízení elektromotoru. Kolik je firem zabývajících se výrobou elektrických pohonů, tolik může existovat různých kombinací řídících prvků. Jednou z těchto firem je firma National Instruments, která vyvíjí jak měřící, řídící a výpočetní hardware, tak i software.

V této práci se budu zabývat řízením krokového motoru za pomoci produktů od firmy National Instruments. Konkrétně se jedná o zařízení CompactRIO, které patří mezi modulární vestavné systémy. Jeho modularita spočívá v přizpůsobitelnosti dané aplikaci jednoduchým přidáním potřebných modulů. Dále pak využívám k řízení systém LabView, což je softwarový prostředek pro realizaci řídícího algoritmu ve spolupráci se zařízením CompactRIO.

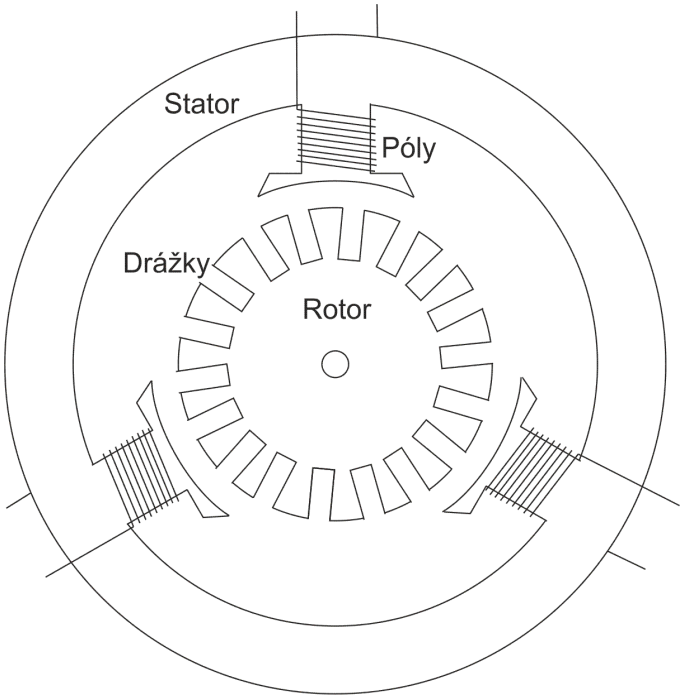
Na začátku této práce popíšu funkci elektromotoru a krokového elektromotoru, okomentuji způsob řízení krokového motoru včetně řídícího signálu, zařízení a polovodičového měniče. Poté se zmíním o softwarovém prostředku LabView. V praktické části práce se budu věnovat zapojení a zprovoznění krokového pohonu. Nakonec představím mnou vytvořený řídící algoritmus v prostředí LabView pomocí grafického programování a grafické uživatelské rozhraní výsledného programu.

2. Teoretická část a technické údaje

2.1 Elektrický motor

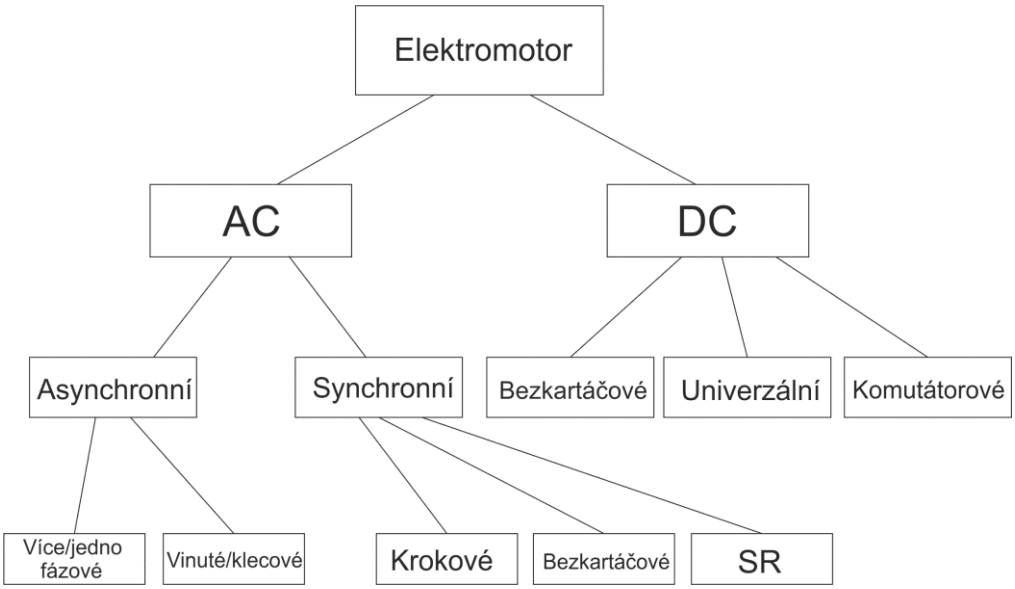
2.1.1 Definice, třídění

Než se budu zabývat krokovým motorem, je třeba definovat, co je to elektrický motor (elektromotor). Elektromotor je elektromechanický měnič energie, který konvertuje energii elektrickou na energii mechanickou. Každý rotační elektromotor se skládá ze dvou hlavních částí – z rotoru a statoru. Stator je část elektromotoru, která zůstává v klidu a nehýbe se. Na statoru jsou připevněny póly, na nichž je navinuto vinutí (zpravidla budící), konce vinutí jsou vyvedeny na svorky. Rotor je částí elektromotoru, která je uložena uvnitř statoru a rotuje (nemusí tomu tak být vždy, existují motory, jejichž rotory obemykají stator zvenku, jedná se zejména o malé motory pro robotiku a modelářství). Rotačního pohybu je u různých typů motorů dosahováno různě, vždy se však jedná o interakci magnetického pole vzniklého průchodem elektrického proudu vinutím statoru a magnetického pole v rotoru (buzeného buď průchodem proudu rotorovým vinutím, nebo permanentním magnetem).



Obrázek 1 - Příklad zjednodušené konstrukce střídavého elektromotoru, stator a uvnitř se točí rotor, v jehož drážkách je uloženo vinutí, Zdroj: autor

Elektromechanická přeměna energie bývá u mnoha strojů reverzibilní, lze tedy aplikací mechanické energie na hřídel točivého elektrického stroje konvertovat mechanickou energii na elektrickou, pak se však jedná o generátor, jímž se v této práci zabývat nebudu.



Obrázek 2 - Diagram dělení elektromotorů, Zdroj: autor

Základní dělení elektromotorů se odvíjí od způsobu napájení, respektive od průběhu napájecího proudu. Motory tedy dělíme na napájené stejnosměrným proudem (DC) a střídavým proudem (AC). Mezi stejnosměrné motory řadíme motory univerzální, komutátorové a částečně také bezkartáčové (BLDC). Stejnosměrné motory nacházely uplatnění v minulých desetiletích ve všech možných aplikacích od spotřebičů po trakci, v poslední době jsou však na ústupu, zejména kvůli horší mechanické kompaktnosti (komutátor je mechanicky namáhán). Střídavé motory pak dělíme na asynchronní a synchronní. Rotory asynchronních motorů se točí asynchronními otáčkami, tj. otáčkami menšími či většími, než jsou otáčky magnetického pole buzeného statorem. Co se týče asynchronních motorů, lze je dále dělit podle počtu fází, podle typu rotoru apod. Asynchronní motory byly na svém vrcholu v posledních letech, zejména díky své spolehlivosti, jednoduché konstrukci a také faktu, že nepotřebují takřka žádnou údržbu. Navíc díky stále novějším a optimálnějším měničům je lze efektivněji i řídit. Tyto motory našly uplatnění ve všech oborech lidské činnosti.

Synchronní stroje se liší od asynchronních tím, že rotor se točí stejnou rychlostí jako magnetické pole buzené statorem. Nejznámější příklad synchronního stroje je turbogenerátor v elektrárně. Pokud jde o motory, jejich uplatnění je stále vyšší díky rozmachu polovodičových měničů. Synchronní motory se ukázaly jako vhodné pro pohony elektromobilů a obecně pro trakci, dále se jedná o menší motory modelářské, pro robotiku, ale i domácí spotřebiče. Mezi synchronní motory patří též spínané reluktanční motory, krokové motory a BLDC motory.

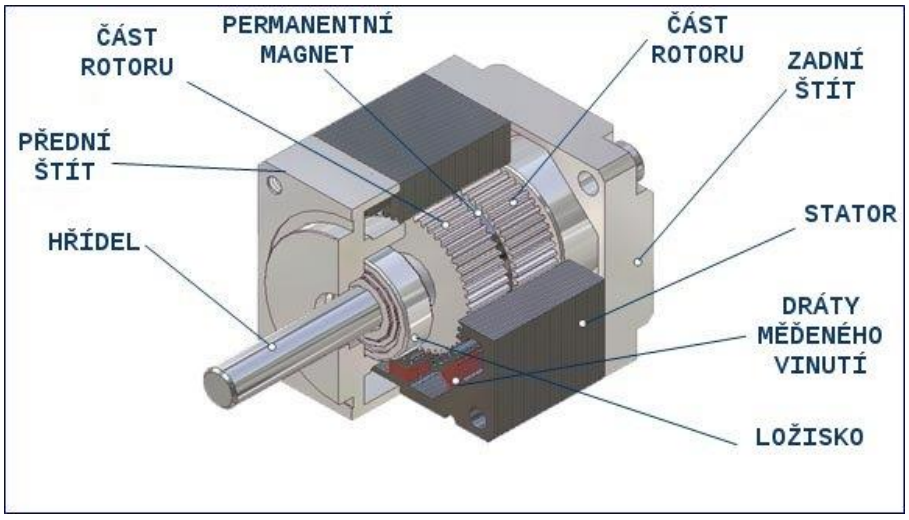
O materiálech využívaných pro výrobu elektromotorů by se dalo napsat mnoho stran, já se pouze zmíním o několika nejdůležitějších. Pro výrobu magnetických obvodů motorů (statorové póly, rotor, póly rotoru) se nejčastěji využívají ocelové plechy legované křemíkem (za účelem snížení ztrát vířivými proudy). Litá nebo kovaná ocel se používá zejména na hřídele a na rotory, v nichž nedochází ke změně magnetického toku (např. turbogenerátor). Kostry elektromotorů bývají ocelové, používá se též litina. Vinutí je nejčastěji vyrobeno z elektrotechnické mědi o vysoké chemické čistotě, menší stroje bývají vinuty jednoduchými žilami, velké stroje mají vinutí vyrobeno z měděných tyčí. Dále je třeba zmínit permanentní magnety, které nacházejí stále větší uplatnění v elektromotorech. Jako materiál k jejich výrobě se používá neodym (NdFeB), samarium-kobalt (SmCo) a AlNiCo. Jednou z nejdůležitějších částí elektromotorů je izolace. Jednak se izolují vodiče (vinutí, přívody) od sebe navzájem a od kostry a magnetického obvodu (u velkých strojů nevodivé pásky, nátěry, laky), dále pak jednotlivé ocelové plechy tvořící magnetické obvody (kvůli ztrátám vířivými proudy) – nejčastěji se používají impregnační laky.

V porovnání s klasickým spalovacím motorem má elektrický motor řadu výhod. Má vysokou účinnost, spolehlivost, krátkodobou přetížitelnost a lze jej dimenzovat na velký rozsah výkonů (mW až MW) a momentů, není potřeba častá údržba. Také nevytváří zplodiny, které by mohly ohrožovat životní prostředí. Jeho hlavní nevýhodou je závislost na okamžité dodávce elektrické energie, nejčastěji ze sítě. Tento problém je však možno částečně či dočasně eliminovat záložním napájením z baterie (v kombinaci s vhodným měničem). Také je dnes již nezbytností u elektrických motorů mít řídící

elektroniku, která leckdy může být dražší než samotný motor. Existují však i dnes některé aplikace elektrických pohonů, které si vystačí s pouhým připnutím na síť a pro svůj běh nepotřebují žádné složité řízení.

2.1.2 Krokový motor

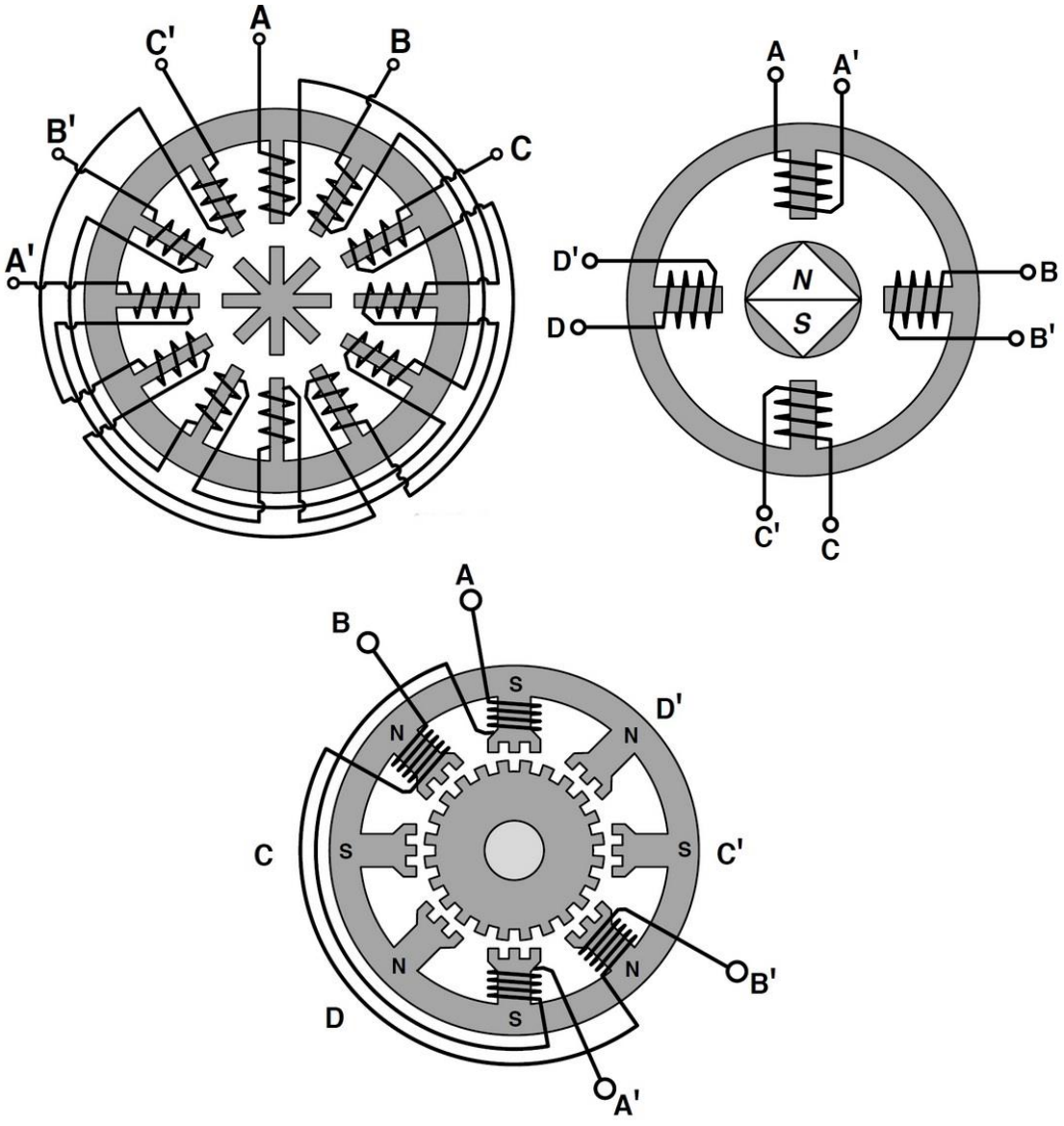
Odtud se již budu věnovat výhradně krokovému motoru, jelikož je předmětem této práce. Pokud bychom chtěli zařadit, do jaké kategorie elektrických motorů patří krokový motor, seznali bychom, že se jedná o motor střídavý, synchronní. Na svorky motoru přivádíme střídavý signál specifického tvaru a rotor se otáčí synchronně s magnetickým polem tvořeným statorem. Pokud nejsou na rotoru permanentní magnety, jedná se o reluktanční motor. Základní princip fungování krokového motoru



Obrázek 3 - Konstrukce hybridního krokového motoru [12]

je založen na pohybu rotoru o přesný úhel – krok jako reakce na proudový impuls na cívce. Točivé magnetické pole vzniká v motoru díky elektronickému spínači, který přivádí na budící cívky statoru proudové impulsy a tím pohání rotor. Hlavní výhodou krokového motoru je, že má nenulový statický moment. Když se motor zastaví, cívky statoru jsou stále napájeny proudem a vzniklé statické magnetické pole drží rotor v dané poloze určitým momentem. Toho se dá využít například ve zdvihacích strojích, kdy je potřeba udržet nějaké břemeno v určité poloze. Naopak pro krokový motor není moc výhodný běh s častou změnou směru otáčení, protože při změně smyslu otáčení teče vinutím větší proud, motor se tedy více zahřívá a vzrůstají ztráty.

Rotory krokových motorů mohou být několikerého typu – pasivní, aktivní a hybridní. Pasivní rotory jsou takové, které samy o sobě nevytvářejí magnetické pole (permanentní magnety), ani nemají namontováno budící vinutí (elektromagnet). Konstrukce pasivního rotoru má speciální tvar s nerovnoměrnou vzduchovou mezerou. Právě proměnlivá vzduchová mezera zapříčiní rozdílné reluktance magnetického obvodu motoru při různém natočení, tzn. rotor se natočí tak, aby měl co nejmenší energii v magnetickém obvodu. Takové motory můžeme nazvat reluktančními.

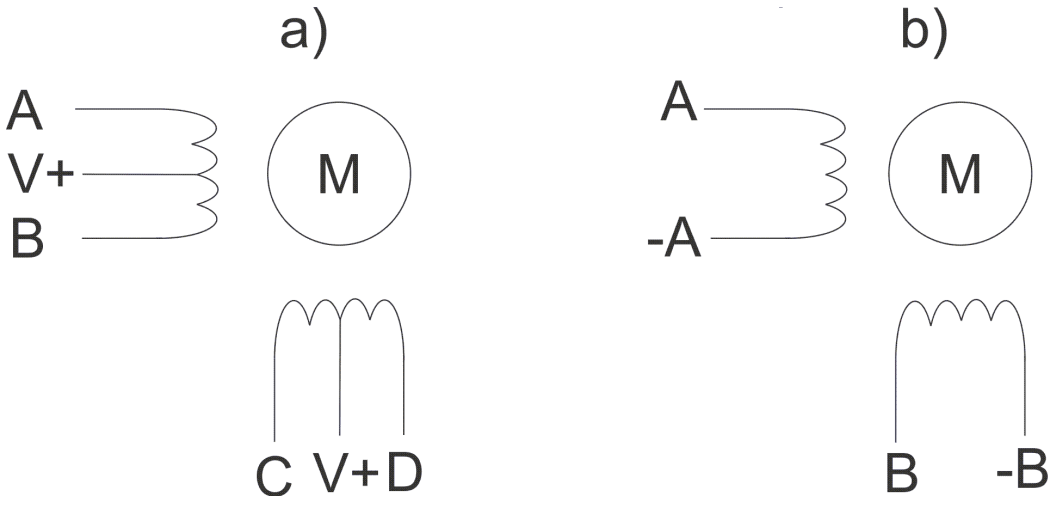


Obrázek 4 - Ukázka konstrukce motoru s pasivním rotorem (vlevo), s aktivním rotorem (vpravo) a s hybridním rotorem (dole) [8]

Tento typ krokového motoru se však v dnešní době již moc nepoužívá. Naproti tomu aktivní rotory mají nalisovány permanentní magnety a točivý moment vzniká

vzájemnou interakcí magnetického pole rotoru a pole vzniklého proudem protékajícím budícími cívkami statoru. Pak se již však nejedná o reluktanční motor. Tento typ motoru lze používat v aplikacích, kde není potřeba dosahovat malých kroků a kde není potřeba velkých výkonů. Rotor hybridních krokových motorů kombinuje výhody obou předchozích typů. Rotor je tvořen axiálním permanentním magnetem, na něj jsou nalisovány dva ozubené póly, které jsou vzájemně pootočeny o polovinu zubové rozteče. Rozložení zubů po obvodu rotoru je nerovnoměrné, aby bylo možné určit, kterým směrem se bude rotor točit a aby vůbec k rotaci došlo.

Dále se krokové motory odlišují zapojením budícího vinutí. Existují motory unipolární a motory bipolární. Zásadní rozdíl je v tom, že v unipolárním zapojení proud v cívce nemění svoji polaritu, lze tedy říci, že na každou cívku je posílán kladný impuls a ke spínání jednotlivých fází by stačil pouhý tranzistor. V bipolárním zapojení se mění polarita pulsu posílaného na cívky, typicky je tedy potřeba zapojení do H-můstku, aby došlo ke komutaci. Na první pohled lze rozeznat tato dvě zapojení snadno – stator zapojený unipolárně má 3 vývody na každou fázi, v bipolárním zapojení má pouze 2 vývody (jak lze vidět na obrázku).



Obrázek 5 - a) Unipolární zapojení vinutí, b) Bipolární zapojení vinutí, Zdroj: autor

2.1.2.1 Krokový motor SX23-1412

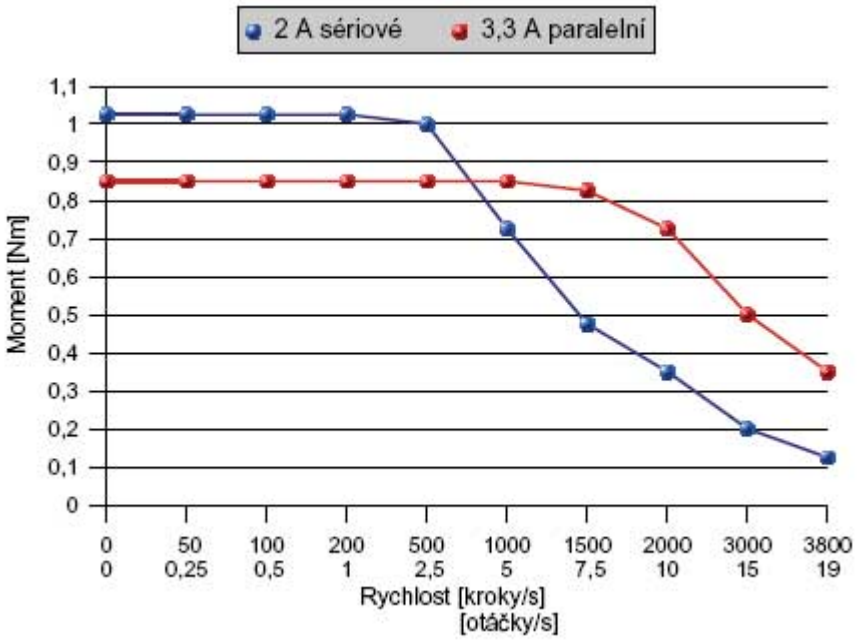
K provedení této práce jsem dostal k zapůjčení krokový motor SX23-1412 od firmy Microcon. Jedná se o dvoufázový motor s hybridním rotorem. Při realizaci této

úlohy bylo vinutí zapojeno bipolárně, tento motor však dokáže pracovat i jako unipolární stroj. Technické parametry a jmenovité hodnoty elektrických a mechanických veličin motoru jsou zaznamenány v tabulce.

Tabulka 1 - Katalogové parametry krokového motoru SX23-1412 [11]

Řady SX - příruba NEMA23						
Typ	**Statický moment (Nm)**	**Jmenovitý proud (A)**	**Indukčnost (mH)**	**Odpor (Ω)**	**Moment setrvačnosti rotoru (kgm2x10^{-3})**	**Hmotnost (Nm)**
SX23-1412	1,2	1,4 / 2,8	10 / 2,5	3,6 / 0,9	0,03	0,7

Na jedno otočení připadá 200 kroků, jeden krok tedy odpovídá otočení o 1,8°±0,1°. Motor lze provozovat i s menším krokem (realizovatelné softwarově), avšak je zde omezená přesnost kroku. Pouze v případě potřeby plynulé rotace při nižších otáčkách lze využít menšího kroku. Krokový motor SX23-1412 je třídy izolace B (130°C).



Obrázek 6 - Momentová charakteristika krokového motoru SX23-1412 [11]

Na obrázku 6 je zobrazena momentová charakteristika krokového motoru SX23-1412, převzatá je z katalogu firmy Microcon. Je zde patrná zvláštní funkce krokového motoru, a to že má statický moment přes 1 Nm při sériovém zapojení budícího vinutí a 0,85 Nm při paralelním zapojení vinutí. Lze také vidět, že moment s rostoucími otáčkami klesá celkem značně, proto se krokové motory nepoužívají tak často v aplikacích, kde je potřeba dosáhnout vysokých otáček (řádově tisíce otáček/min).

Motor lze řídit ve dvou módech – krokově, nebo klasicky plynule. Když se dosáhne určitých otáček, motor sám již nedokáže krokovat a přechází v plynulou rotaci. Plynulejší rotace lze dosáhnout i při nižších otáčkách, je však potřeba snížit velikost kroku.

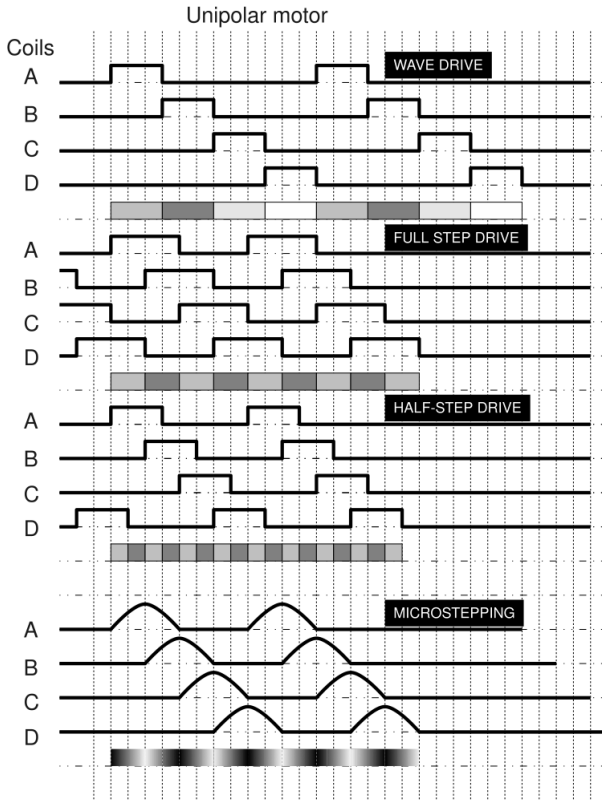
2.2 Řízení motoru

Krokové motory patří do skupiny spínaných motorů (též elektronicky komutovaných), jelikož jsou napájeny proudovými signály z elektronických spínačů, přesněji řečeno měničů. Elektronický spínač je napájen stejnosměrným proudem (většinou ze spínaného zdroje), každá fáze motoru je připojena na jeden elektronický spínač. Na elektronické spínače jsou poté přiváděny řídící pulsy, které v určitém pořadí a délce pouští proud do budících cívek motoru. Je tedy potřeba nějakým způsobem generovat řídící signál a přivést ho ve správném pořadí a ve správném čase na správné spínače.

2.2.1 Řídící signál

Jak již bylo zmíněno, krokový motor je řízen vhodným střídavým připínáním proudových impulsů na cívky statorového vinutí. Z hlediska řízení lze krokové motory dělit na jednofázové a vícefázové. Počet fází znamená, kolik párů protilehlých pólů je na statoru realizováno. Obecně platí, že čím více fází, tím jemnějšího krokování lze dosáhnout.

K řízení motoru je tedy potřeba realizovat signál, který bude periodicky posílat impulsy na jednotlivé cívky statoru – PWM (Pulse-Width Modulation) signál. PWM signál lze generovat mnoha způsoby (programově, mikroprocesorově, hardwarově). Abychom vytvořili točivé magnetické pole, musíme spínat cívky tak, aby se střídaly jednotlivé fáze. Obecný příklad takového PWM signálu lze vidět na následujícím obrázku.



Obrázek 7 - Příklad průběhů proudových pulsů připojených na jednotlivá vinutí unipolárního krokového motoru [10]

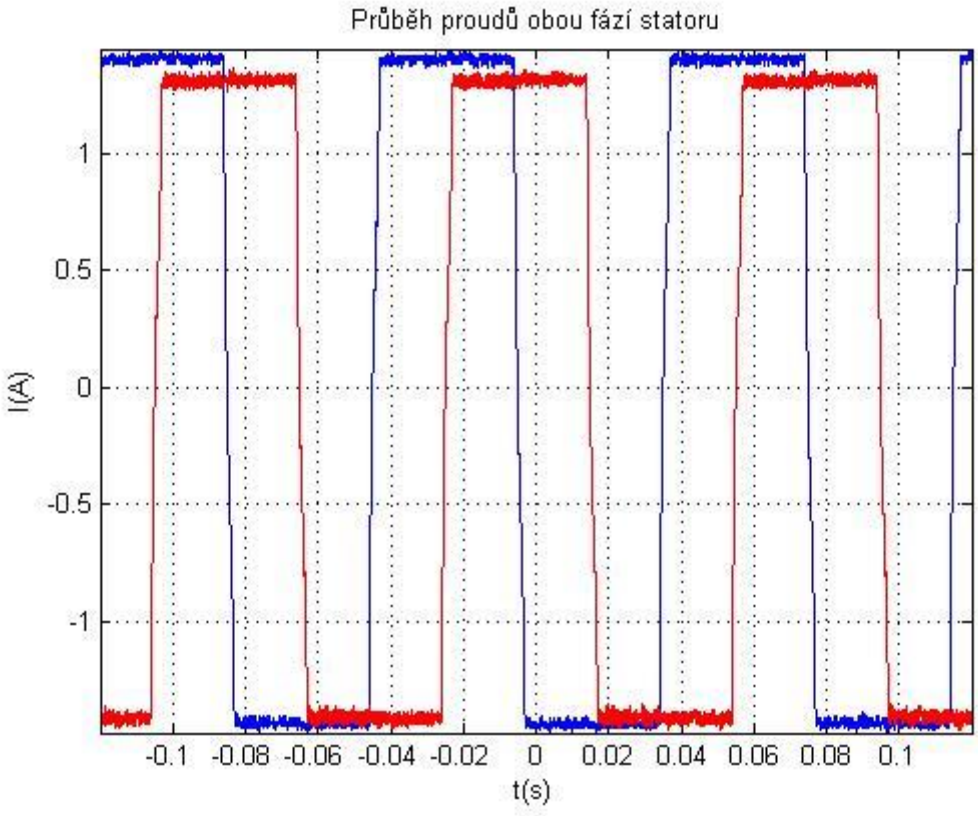
V první části obrázku je vidět průběh proudových pulsů budících vinutí, kdy je buzena vždy jen jedna cívka. Tento způsob řízení se moc často nepoužívá, není totiž dosažen maximální možný moment. V druhé části je znázorněno řízení při plném kroku, kdy se pulsy na jednotlivé cívky překrývají v polovině sepnuté doby. Tento způsob řízení zajišťuje maximální využitelný moment motoru. Na posledních dvou grafech jsou průběhy při zmenšeném kroku, lze tak dosáhnout větší přesnosti, klesá však využitelný moment motoru.

Generovat PWM signál k řízení elektronického měniče lze mnoha způsoby. Nejjednodušší způsob je analogově za pomoci trojúhelníkového signálu (generovaného například v integrátoru) a komparátoru. Komparátor porovnává hodnotu trojúhelníkového signálu se stejnosměrným signálem a při každém průniku překlápí logickou hodnotu na výstupu. Velikostí stejnosměrného signálu lze tedy měnit střídu výstupního signálu, změnou frekvence trojúhelníkového signálu lze měnit periodu PWM. Dále je možné generovat PWM signál digitálně pomocí mikroprocesoru

(využitím timeru a compare unit nebo dvou timerů) a poté signál převést na analogový přes AD převodník. Spousta firem již však dnes nabízí moduly či integrované obvody, které dokáží flexibilně generovat PWM signál, aniž by člověk musel znát podrobné vnitřní zapojení. V této práci jsem využil produktů firmy National Instruments, které mají vysoký stupeň integrace, takže v jednom modulu je zakomponována jak řídící elektronika, tak měnič a řídící počítač. Samotné řízení je pak otázkou software, kterému se budu věnovat v této práci později.

2.2.2 Naměřené průběhy

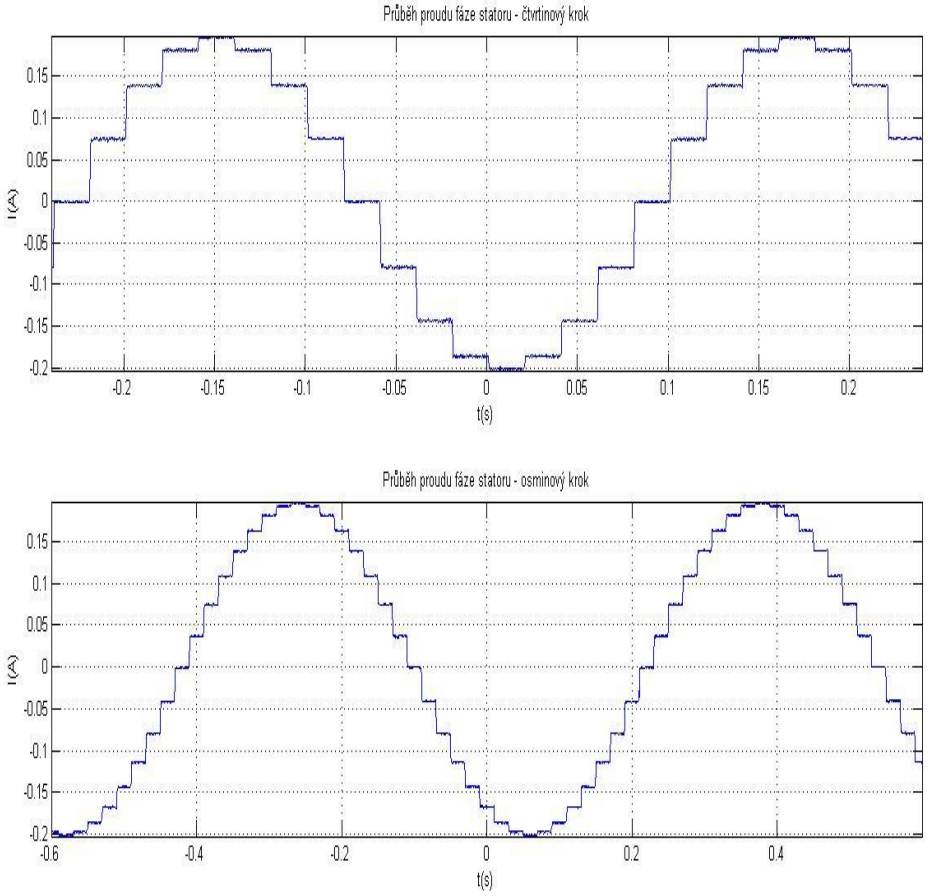
Bipolární hybridní krokový motor je napájen střídavým obdélníkovým signálem. Tento signál jsem zaznamenal pomocí osciloskopu. Průběh signálů napájejících obě fáze motoru lze vidět na následujícím obrázku.



Obrázek 8 - Průběhy proudů fází statoru bipolárního krokového motoru, Zdroj: autor

Lze vidět, že obdélníkové průběhy proudů obou fází jsou navzájem posunuty o čtvrtinu periody, jedná se zde o tzv. čtyřtaktní řízení s plným krokem. Na dalších obrázcích jsou zaznamenané průběhy proudů fází při zmenšeném kroku (čtvrtinovém

kroku a osminovém). Lze vypozorovat, že při zmenšeném kroku (tzv. mikrokrokování) přechází řízení z obdélníkové komutace na komutaci sinusovou. Toho lze využít jednak pro přesnější zaměřování polohy, pro plynulejší rotaci při nižších otáčkách, avšak také pro dosažení vyšších otáček. Při vysokých otáčkách (naměřeno 300 ot. /min) již obdélníková komutace nestačí a může se stát, že krokový motor nestíhá krokovat, sinusová komutace je však plynulejší a rotor tak snáze krokuje i při vyšších rychlostech. Se zapůjčeným vybavením lze dosáhnout úrovně mikrokrokování až 1/256.



Obrázek 9 - Průběhy proudů jedné fáze při mikrokrokování, Zdroj: autor

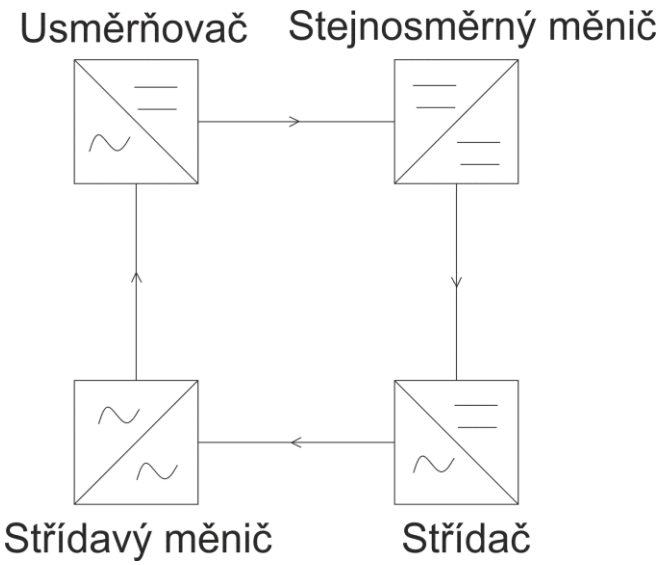
2.2.3 Elektronický měnič

Měnič elektrické energie je v principu zařízení, které mění elektrickou energii na elektrickou energii jiných parametrů. Měniče mohou transformovat stejnosměrný proud na střídavý (střídače) nebo naopak (usměrňovače). Pak existují měniče, které mění stejnosměrný proud na tentýž, jen o jiné velikosti (‚chopper') a v neposlední řadě se využívají měniče měnící parametry střídavého proudu (často se jedná o tzv.

frekvenční měniče). Všechny tyto typy měničů se dnes vyrábí za pomoci polovodičové techniky, resp. polovodičových součástek.

Polovodičové měniče jsou pro řízení motorů zcela zásadní, bez nich je řízení značně omezené (řídit lze např. počtem pól-párů, odporovými spouštěči, stykači apod.). Měniče tedy dovolují řídit motory plynule a v širokém rozsahu parametrů elektrického napětí/proudu či momentu.

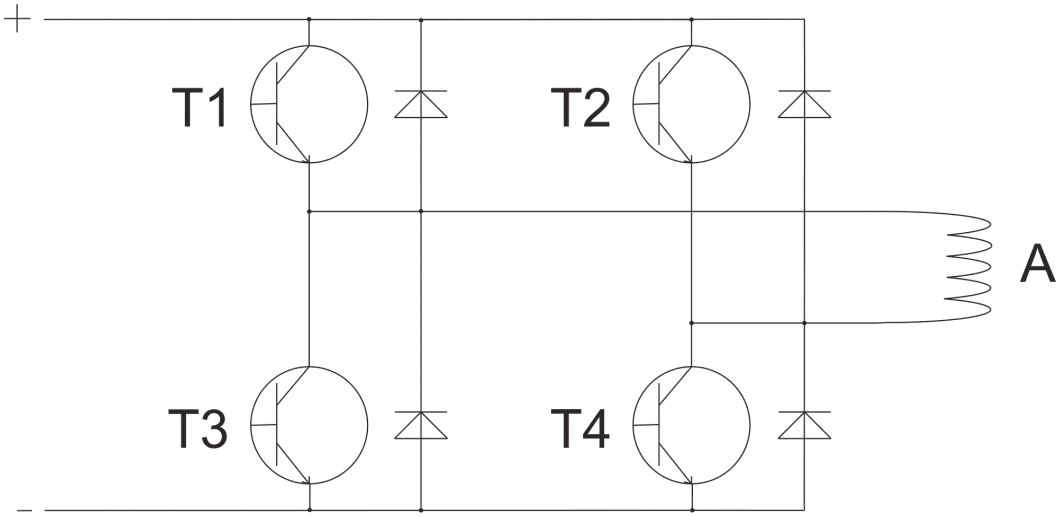
Měniče jsou tedy elektrické obvody složené z několika různých polovodičových prvků/součástek. Princip jejich fungování je takový, že řídící část o nízké úrovni napětí a proudu spíná a ovládá silovou část obvodu. Typickou polovodičovou součástkou, která se využívá v měničích a zároveň je tou nejjednodušší, je dioda. Jedná se o pouhý PN přechod zapojený do obvodu, který má za účel propouštět proud pouze jedním definovaným směrem, což je dáno charakteristikou materiálu na styku polovodiče typu P a N. Další součástkou hojně používanou je tyristor.



Obrázek 10 - Diagram zobrazující typy elektrických měničů, Zdroj: autor

Tyristor má podobnou funkci jako dioda, avšak jedná se již o řízenou součástku. Tyristor tedy propouští proud tehdy, když je na jeho hradlo přiveden obdélníkový proudový impuls. Tyristory mívají uplatnění ve výkonových aplikacích ve velkých usměrňovačích, jsou vyráběny ve všech možných konfiguracích (s chladiči apod.). Poslední součástkou, o které se zmíním, je tranzistor. Tranzistor je součástka se dvěma

PN přechody, existují tedy dva typy tranzistorů – NPN a PNP. Jejich hlavní výhodou je skutečnost, že dokáží zesilovat signál, využívají se tedy hodně v komunikacích a elektronice. Výkonové tranzistory mají hlavně za úkol jednu věc – při přivedení napětí na gate (hradlo) tranzistoru sepnout výkonovou část obvodu. Ve výkonové elektronice se v poslední době často využívají vypínatelné součástky IGBT – spojení unipolárního a bipolárního tranzistoru, které dovoluje spínat obvody o napětí v řádech kilovoltů a proudech v řádech kiloampérů.



Obrázek 11 - Schéma tranzistorového H-můstku: T1-T4 jsou spínací tranzistory, A je cívka statorového vinutí, Zdroj: autor

V elektrických pohonech bývá časté zapojení měničů do můstku. Podle počtu fází motoru pak určíme, kolik je potřeba polovodičových součástek. V případě dvoufázového krokového motoru tedy potřebujeme pro každou cívku jeden měnič se 4 polovodičovými prvky. Na předchozím obrázku je vyobrazen měnič zapojený do můstku.

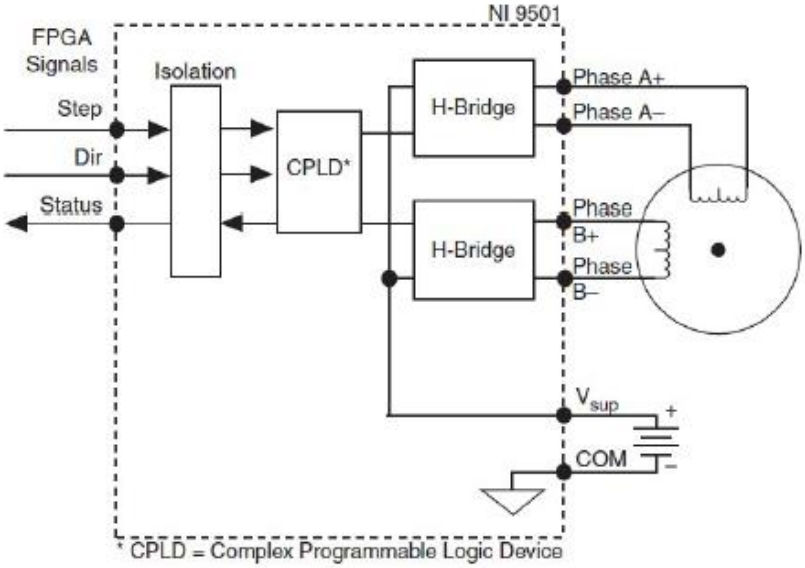
Tento měnič (na obrázku 11) je napájen stejnosměrným proudem. Vždy jsou pak spínány protilehlé tranzistory, v tomto případě tedy dvojice T1, T4 a T2, T3. Tímto způsobem dochází k střídání polarity proudu procházejícího přes cívku vinutí krokového motoru (A). Každý tranzistor má svoji paralelně zapojenou komutační diodu, která při přepínání z jedné dvojice tranzistorů na druhou svým napětím umožní vypnutí daného tranzistoru. Ve své podstatě se tedy jedná o střídač, protože výstupní proud

má střídavý charakter, což je průběh, který u bipolárního krokového motoru potřebujeme.

Měničů existuje velké množství, lze sestavit střídače, frekvenční měniče, jednopulsní, vícepulsní usměrňovače, lze je zapojovat můstkově, využívat u nich různé filtrační a komutační pomocné obvody. Dá se říci, že každá firma má svoji konfiguraci součástek a je jen na zákazníkovi, jaké vyžaduje parametry měniče.

2.2.4 Modul NI 9501

Tento modul, vyvinutý firmou National Instruments, je přímo určený k řízení krokových motorů a je kompatibilní se všemi možnými systémy (především LabView od National Instruments, dále pak např. Matlab). Jedná se o již zmíněnou formu měniče, kdy je v jednom bloku integrován jak výkonový měnič, tak generátor impulsů a logické řídící obvody. Modul je připojen k napájecímu zdroji motoru a k motoru samotnému. Modul NI 9501 funguje jako výkonový měnič a spínač, který po přijetí informace od programu vysílá impulsy na fáze motoru. Programově je tedy potřeba jen nastavovat 2 parametry – Step (povel k vykonání kroku) a Direction (určení směru krokování), parametr Status podává informaci i napájení modulu. Vnitřní zapojení je blokově znázorněno na následujícím schématu:



Obrázek 12 - Schéma vnitřního zapojení modulu NI 9501 [6]

Každá fáze motoru má tedy svůj H-můstek, který řídí přívod pulsů na cívky statoru. Můstky jsou komplexně řízeny v CPLD jednotce, která získává povely parametrů Step a Direction přes procesor/FPGA z počítače. Modul má svorkovnici s 10 svorkami – 4 svorky pro póly fází motoru, svorku pro kladný potenciál, svorku pro zem, svorku pro ochranný vodič a 3 svorky jsou nepřipojeny.

Obrázek 13 - Fotografie modulu NI 9501 [6]

2.2.5 CompactRio

Dosud jsem se zabýval řídícím signálem, měničem, modulem, ve kterém jsou tato zařízení implementována, zbývá tedy popsat, čím je to vše řízeno a synchronizováno s počítačem. Zařízení CompactRIO 9076 patří do skupiny přístrojů cRIO firmy National Instruments. Toto rozhraní umožňuje širokou aplikaci zejména ve vestavných průmyslových a vědeckotechnologických systémech díky své snadné modifikovatelnosti a kompaktnosti. CompactRIO 9076 je šasi se zabudovaným Real-time procesorem, kontrolérem, programovatelným hradlovým polem (FPGA) a 4 sloty pro přídavné moduly. Bližší specifikace uvádím v následující tabulce:

Tabulka 2 - Katalogové údaje zařízení CompactRIO 9076 [4]

Počet slotů	Frekvence procesoru	FPGA	DRAM	Interní paměť	Ethernet port	Serial port	USB port	Zdroj napájení
4	400 MHz	Spartan 6 - LX45	256 MB	512 MB	Ano	Ano	Ano	9 - 30 VDC

Toto zařízení je používáno jako základní deska, na níž se připevní moduly od firmy National Instruments plnící konkrétní funkce (řídící, měřící moduly, pulsní generátory, analyzátory atp.) – v našem případě je to modul NI 9501. CompactRIO je připojeno k počítači přes rozhraní TP-Link, sériový port nebo přes USB. CompactRIO je napájeno ze spínaného zdroje hodnoty napětí 24 VDC optimálně, celkový povolený rozsah činí 9 – 30 VDC.

Obrázek 14 - Fotografie zařízení CompactRio 9076 [4]

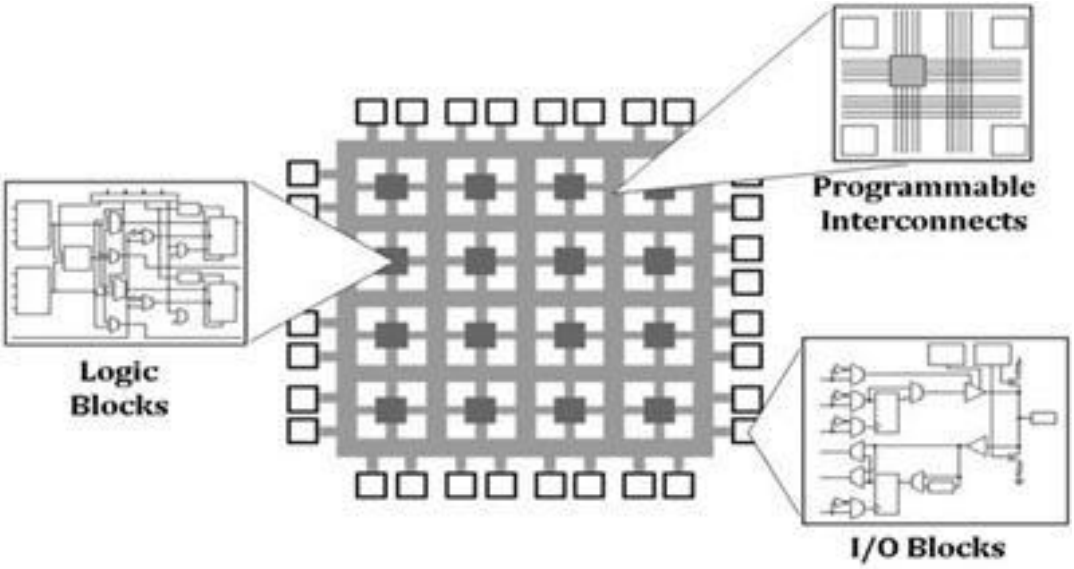
CompactRIO zde slouží jako prostředník mezi počítačem a motorem. Z počítače je do mikroprocesoru v CompactRIO uložen program, který řídí měnič a motor. Pro rychlé výpočty, synchronizaci a tvorbu řídících signálů je využito programovatelné hradlové pole, které je rovněž zakomponováno do základní desky. Rovněž je zde implementován externí zdroj časování o frekvenci hodin 40 MHz, který lze využít k synchronizaci dílčích částí programu či přesnému generování řídících signálů.

2.2.6 FPGA

Na šasi zařízení CompactRIO, jak již bylo zmíněno, je namontováno také hradlové pole. Programovatelné hradlové pole (dále jen FPGA) je mikročip s vysokým stupněm integrace. Na rozdíl od mikroprocesoru nemá pevně definovanou vnitřní strukturu logických obvodů, je plně modifikovatelné. Jedná se tedy o pole jednoduchých logických obvodů, síť vodivých propojek a vstupních a výstupních vývodů. Počet logických obvodů na čipu se pohybuje v řádu miliónů, je zde tedy extrémní flexibilita ve vývoji aplikací pro FPGA. Co se týče základních logických obvodů, bývají to obvody AND (logický součin), NAND (negovaný logický součin), OR (logický součet), XOR (exkluzivní logický součet), ze složitějších pak různé sčítačky, odčítačky apod. FPGA se používají tam, kde není vhodné použít mikroprocesor nebo na menší série výrobků, pro které by nebylo ekonomicky výhodné vyvíjet a vyrábět jednoúčelový procesor.

Jednotlivé programy, výpočty a funkce se na FPGA realizují propojováním logických obvodů, rozsah programu je tedy dán hlavně velikostí hradlového pole. Díky jedinečné struktuře lze také provádět výpočty paralelně, na rozdíl od běžného mikrokontroléru, FPGA tedy vynikají svojí rychlostí. Potřebují však externí paměť, protože po zapnutí se musí FPGA nakonfigurovat.

Programy, které jsou nahrávány do FPGA, vytváříme v počítači pomocí speciálního jazyka pro popis hardware – VHDL (VHSIC-Hardware-Description-Language). Další možností je grafické programování (LabView – umožňuje vysokou úroveň abstrakce) nebo někteří výrobci umožňují programovat pomocí schématického zobrazení zapojení logických obvodů (ISE Webpack). Takto naprogramované kódy jsou pak přeloženy do konfiguračního souboru FPGA v určitých krocích (syntéza, mapování, rozmístění a propojení, generování konfiguračního souboru).



Obrázek 15 - Schématické zobrazení FPGA s popisem jednotlivých částí [9]

2.3 Software

Od hardwaru se nyní přesunu k softwaru. Elektrický pohon je sice řízen mikrokontrolérem, popřípadě za pomoci FPGA, je však nutné tyto komponenty propojit a naprogramovat, což lze uskutečnit nejsnáze za pomoci osobního počítače. Existuje nepřeberné množství vývojových prostředí k programování mikroprocesorů a FPGA, které pracují na různé úrovni abstrakce psaného kódu. Jako příklad bych uvedl program MPLab IDE od firmy Microchip, v němž lze programovat za pomoci jazyka C, nebo ISE Webpack od firmy Xilinx, v němž lze pracovat graficky s logickými obvody (primárně je využíván k programování FPGA). V této práci však pracuji se sadou programů od firmy National Instruments – LabView a jeho doplňkovými moduly.

2.3.1 LabView

LabView je softwarové vývojové prostředí od firmy National Instruments, pracující na velmi vysokém stupni abstrakce. Jedná se o mocný nástroj, jehož využití má mnoho podob. Lze jej využít pro měření, zaznamenávání a analýzu dat, tvorbu pokročilých algoritmů, jakožto i k řízení hardwarových zařízení a tvorbu vestavných (embedded) systémů. Práce v tomto prostředí je velmi intuitivní, avšak také velmi funkční a rychlá, vývojář tedy netráví čas luštěním psaného kódu a může se více zaměřit na samotný problém. LabView funguje na principu tzv. grafického

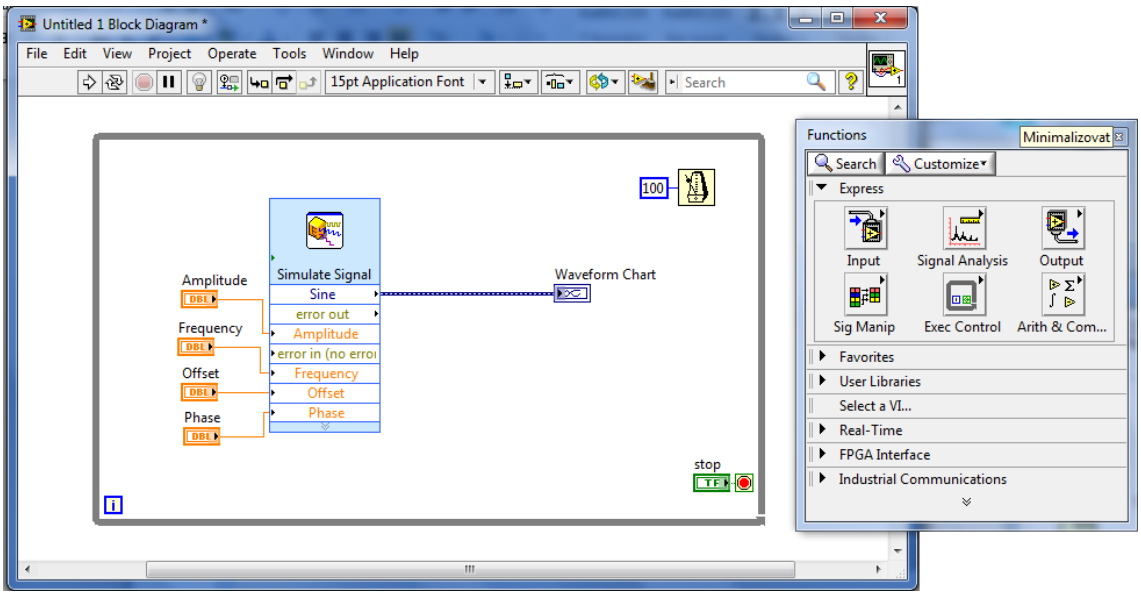
programování, program je tvořen graficky (vkládáním funkčních bloků a spojováním bloků v logické celky), ne psaním kódu. Uživatel tak vytvoří program, který lze pak exportovat jako samostatně fungující aplikaci. V této práci jsem pracoval s plnou verzí LabView 2012 SP1 se školní licencí.

Soubory, které vytváříme v prostředí LabView, se nazývají VI (Virtual Instrument). VI jsou dílčí části, z nichž lze sestavit finální aplikaci, lze však vytvořit i aplikaci z pouhého jednoho VI. Každé VI je tvořeno ve dvou oknech, tzv. Block diagram a Front panel. Okno Block diagram slouží k vytvoření jádra programu, zde jsou vkládány funkce. V okně Front panel vytváříme hlavní panel naší aplikace – grafické rozhraní. Zde vkládáme ovládací prvky (*Controls*), indikátory (*Indicators*), výstupní grafy apod. Front panel tedy ukazuje, jak bude výsledná aplikace vypadat. Nyní se podíváme na jednotlivá okna podrobněji.

2.3.1.1 Block diagram

Block diagram je hlavní okno, v němž realizujeme potřebný kód. Jak již bylo zmíněno, jedná se o grafické programování, principiálně tedy probíhá tak, že do Block diagramu vkládáme funkční bloky a spojujeme je. Na hlavní liště máme klasickou nabídku, kterou známe z OS Windows s přidanými záložkami ‚*Project*' a ‚*Operate*'. Pod hlavní lištou se nachází lišta, která obsahuje důležité funkce pro práci s kódem a pro odstraňování chyb. V okénku *Functions* máme na výběr širokou škálu funkčních bloků a struktur (nazývají se souhrnně *Functions*), které lze v programu využít. Jako příklad uvedu skupinu *Programming*, v níž jsou obsaženy všechny možné programovací nástroje a struktury jako jsou cykly, proměnné, matematické funkce, booleovské funkce, pole atp. V nabídce *Functions* však rovněž najdeme bloky pro časování, pro komunikaci s hardwarem, měření, záznam a zobrazení dat.

Samotné programování je založeno na takzvaném ‚toku dat' (Data-flow). V textových programovacích jazycích je kód interpretován postupně po řádcích, zde je interpretován v definovaném směru a to zleva doprava. Je proto žádoucí tento směr zachovat (jednak pro přehlednost kódu, jednak pro optimální využití možností LabView).



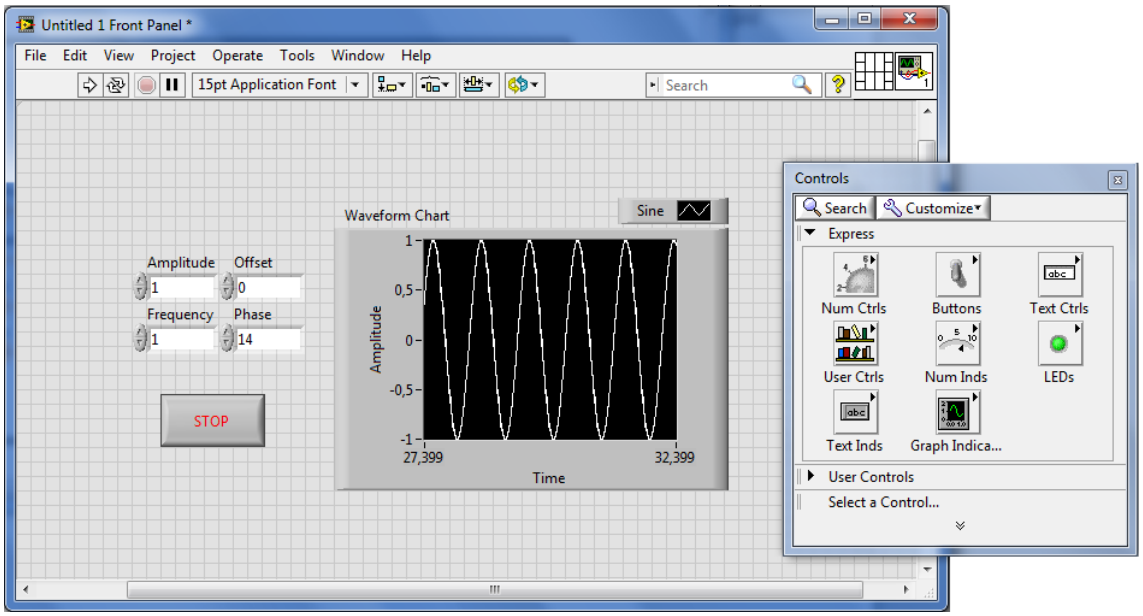
Obrázek 16 - Screenshot jednuduchého programu v Block diagramu, Zdroj: autor

V LabView existuje několik programovacích stylů, podobně jako v ostatních jazycích. Pro aplikace, které mají proběhnout jednou a zobrazit výsledek (např. kalkulátor), stačí vložit bloky a spojit je (výstupy vstupních bloků na vstupy výstupních bloků). Pro aplikace běžící kontinuálně je potřeba implementovat nějaký cyklus (*for*, *while* apod.) a v něm realizovat kód. Složitější strukturou může být stavový automat, který sestává z *while*-cyklu, struktury *switch* a posuvného registru. Na obrázku 16 je příklad jednoduchého programu, kde je smyčka *while*, v níž se simuluje sinusový signál a zobrazuje se pomocí funkce *Waveform chart*. U bloku *Simulate signal* lze upravovat několik atributů signálu (amplitudu, frekvenci, fázi, offset). Ve smyčce je ještě blok pro časování (určuje dobu jednoho běhu smyčky) a *Control Stop*, který program ukončuje. Programy vytvořené v LabView lze kdykoliv ukončit i tlačítkem na liště, avšak není to legitimní způsob ukončování aplikace, měl by se tedy používat pouze v případě nouze (např. vytvoření nekonečného cyklu), rozhodně by neměl sloužit ke standardnímu ukončení programu.

V okně Block diagram vkládáme tedy *Controls* a *Functions*. To jsou funkce a proměnné, jejichž hodnoty lze měnit v reálném čase při spuštění programu z okna Front panel, o němž bude další část této práce.

2.3.1.2 Front panel

Front panel je druhé pracovní okno v LabView. Ve Front panelu vytváříme grafické rozhraní naší aplikace. Vždy, když v Block diagramu vytvoříme nějaký objekt, jehož atributy lze měnit, objeví se grafická reprezentace tohoto objektu ve Front panelu. Programujeme-li tedy aplikaci v LabView, výsledný program bude mít vzhled takový, jaký vytvoříme ve Front panelu. Toto okno má opět hlavní lištu podobnou té z OS Windows, pod ní je uložena lišta určená k ovládání běhu námi vytvořené aplikace.



Obrázek 17 - Screenshot okna Front panel, v němž vidíme grafické rozhraní programu vytvořeného v Block diagramu na předchozí stránce, Zdroj: autor

V okénku *Controls* máme na výběr z široké škály nástrojů, ovládacích prvků, zobrazovacích prvků, zadávacích oken, kontejnerů a dekorativních prvků, které lze využít pro vytvoření grafického rozhraní. Na obrázku 17 vidíme příklad grafického rozhraní vytvořeného ke kódu z předešlé stránky. Vlevo lze spatřit tzv. *Numeric controls* (numerické kontrolky) – to jsou numerické proměnné, které lze nastavovat v reálném čase, v tomto případě nastavují parametry zobrazované funkce sinus. Napravo vidíme *Waveform Chart*, časový graf, který zobrazuje průběh funkce sinus s námi nastavovanými parametry. Nakonec zde máme také tlačítko *Stop*, které slouží ke správnému ukončení aplikace.

Při tvorbě grafického rozhraní je potřeba dbát na některé zásady, poněvadž sebelepší Block diagram s nepřehledným či nečitelným Front panelem nedokáže správně fungovat. Je nutné, aby byly všechny objekty popsané, logicky roztříděné a rozložené po pracovní ploše, aby se nepřekrývaly atp. Jelikož Front panel je ta část aplikace, s níž pak zákazník či pracovník operuje a ovládá jí, je potřeba, aby byla lidsky srozumitelná a neumožňovala uživateli, aby dostal program do neočekávaných stavů, měl by být tzv. ‚*User-friendly*' (uživatelsky přátelský).

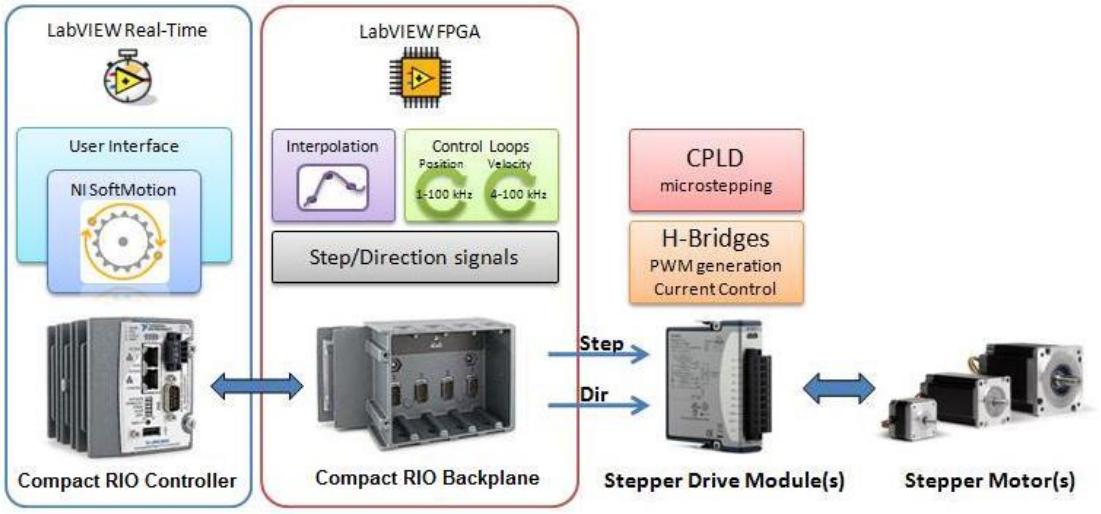
2.3.2 FPGA a Real-Time moduly

V LabView lze pracovat s nepřeberným množstvím zařízení a rozhraní. Jelikož dokáže spolupracovat i s hardwarem od jiných výrobců, než je National Instruments, LabView umožňuje doinstalovat pro potřebný hardware či software doplňující moduly pro práci s daným zařízením. Existují moduly pro práci s měřícími systémy (DAQ – Data-AcQuistion), pro přesné řízení některých typů motorů (Softmotion module), různé analytické kalkulátory (Signal Express module) apod. V této práci jsem využil dva moduly nutné pro práci se zařízením CompactRIO a pro řízení motoru – FPGA module a Real-Time module.

FPGA module, jak napovídá název, umožňuje práci s programovatelnými hradlovými poli. Je k němu ještě potřeba nainstalovat kompilační sadu programů od firmy Xilinx, aby bylo možné na FPGA daný program nahrát. Tento modul otevírá nové funkce v Block diagramu, především vstupně-výstupní uzly pro komunikaci s FPGA nebo tzv. *Single-cycle timed loop*, což je smyčka, která proběhne přesně za dobu jednoho tiknutí vnitřních hodin FPGA (což umožňuje velmi přesné časování a rychlé vykonávání příkazů). Na FPGA lze přímo tvořit VI, kompilátory samy přeloží kód, aby byl pro FPGA čitelný. To však s sebou nese jisté restrikce. Některé funkce neumožňuje modul FPGA používat, jednak kvůli přílišnému zatížení čipu, některé pouze nejsou kompatibilní s jinými funkcemi (např. nelze vložit funkci, jejíž vykonání by trvalo déle, než jeden tik hodin FPGA, do *Single-cycle timed loop*). Nelze například použít funkci dělení, protože FPGA pracuje pouze s celými čísly a dělením vzniká potenciální riziko

vzniku racionálního čísla. Programování FPGA je však díky tomuto modulu velmi snadné oproti jiným vývojovým prostředím s nižší úrovní abstrakce.

Další modul, který jsem použil k programování řízení motoru je Real-Time module. Tento modul slouží k ovládání zařízení v reálném čase. Umožňuje tedy měnit parametry měření či řízení s minimálním zpožděním, z čehož těží především řídící aplikace pohonů či celých strojů, řízených na základě okamžitých požadavků operátora či změn hodnot. Real-Time module je také nezbytný pro práci se zařízením CompactRIO. Na následujícím diagramu je vyobrazena architektura řídícího programu krokového motoru s použitím modulu Real-Time a FPGA.



Obrázek 18 - Diagram komunikace zařízení [6]

3. Praktické provedení

V této kapitole se budu věnovat mému řešení problému řízení krokového motoru pomocí CompactRIO v LabView. Nejprve popíšu elektrické zapojení a propojení všech zařízení, následně pak instalaci a zprovoznění komunikace všech zařízení. V další části představím řídící algoritmus vytvořený v LabView. Nakonec okomentuji vytvoření uživatelského rozhraní a zformuluji pokyny pro správné ovládání programu.

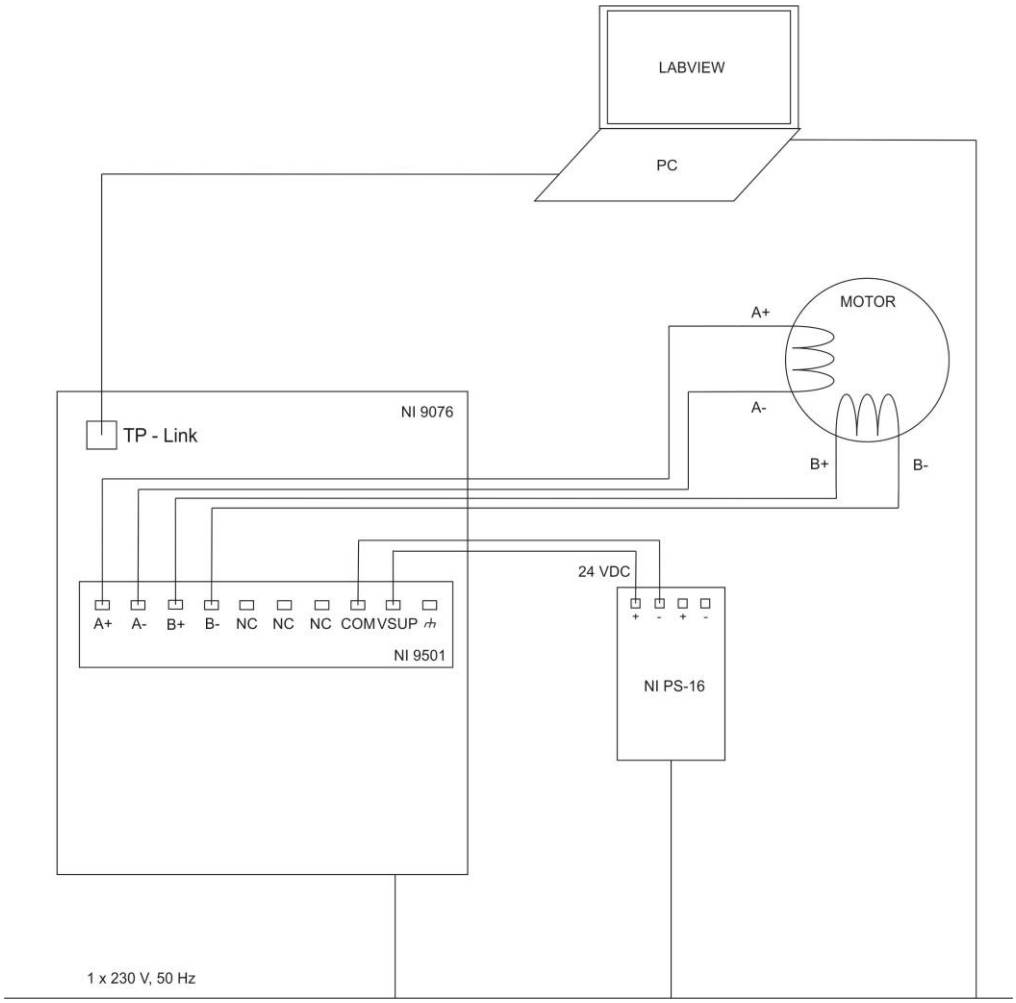
3.1 Instalace systému CompactRIO

V této podkapitole se krátce zmíním o propojení všech zařízení a softwaru potřebných ke správnému nastavení a chodu zařízení. Rovněž zde popíšu konfiguraci zařízení a zprovoznění komunikace.

3.1.1 Celkové schéma

Hardwarová část pohonu krokového motoru obsahuje 5 částí. Je zde krokový motor, modul NI 9501, šasi NI 9076, zdroj napájení pro motor, CompactRIO a počítač. Na obrázku 19 je blokové schéma zapojení celého pohonu.

Počítač je propojen s CompactRIO pomocí ethernetového kabelu, který zajišťuje komunikaci s procesorem a FPGA. Do šasi CompactRIO je zapojen modul NI 9501, který slouží k řízení krokového motoru. Svorky A+, A- jsou připojeny na jednu fázi vinutí, svorky B+, B- jsou připojeny na druhou fázi vinutí. Svorky VSUP a COM jsou připojeny k přídavnému spínanému zdroji 24 V, který napájí krokový motor. Počítač, CompactRIO a zdroj jsou napájeny ze sítě 230 V. Silové obvody (propojení modulu s motorem a se zdrojem) jsou provedeny izolovaným drátem opatřeným dutinkami, připevnění do svorkovnice je šroubkové.



Obrázek 19 - Schéma celkového zapojení krokového pohonu, Zdroj: autor

3.1.2 Software

Pro využití systému CompactRIO je potřeba na počítač nainstalovat několik softwarových nástrojů od firmy National Instruments, které lze buď s omezenou licencí stáhnout na jejich webových stránkách, nebo zakoupit. Já využívám školní studentskou licenci, která zahrnuje veškeré možné prvky a moduly. Následuje seznam potřebných aplikací či souborů:

- LabView (2012/2013)
- NI MAX
- FPGA Module
- Real-time Module
- Xilinx Compilation Tools 13.4
- Device drivers – FPGA, Real-time, CompactRIO

Všechny položky seznamu bylo potřeba nainstalovat, abych zařízení uvedl do provozu. Instalace celého systému může zabrat i několik hodin.

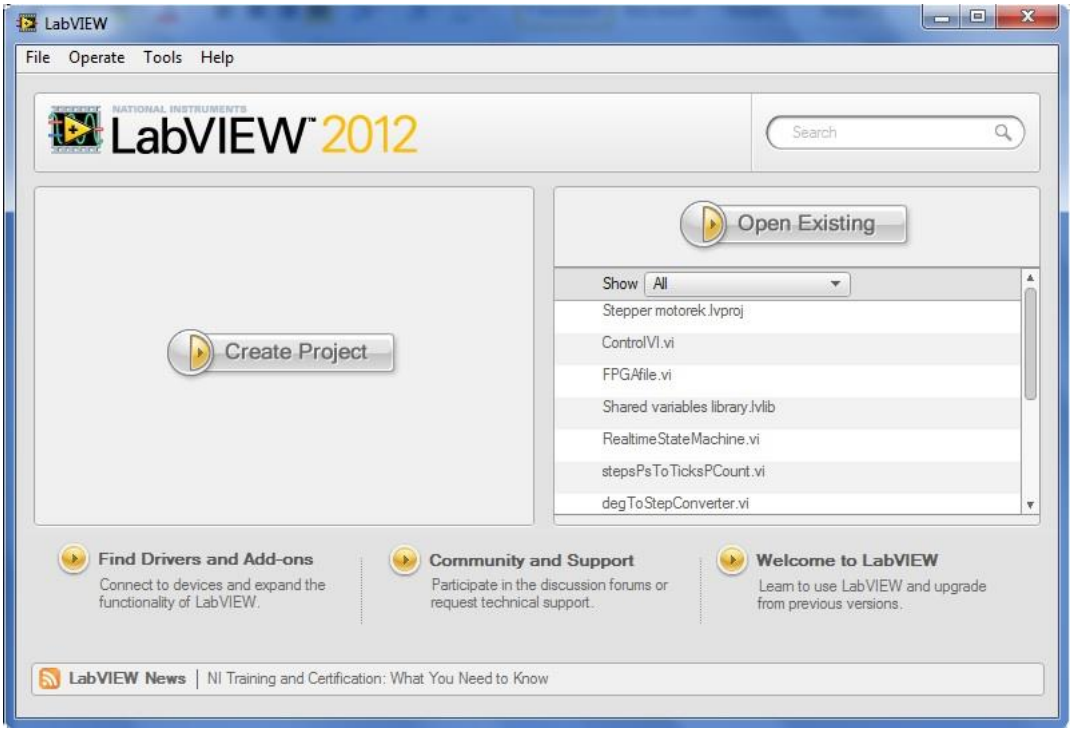
LabView je program, v němž jsem vytvořil řídící algoritmus a grafické rozhraní, použil jsem verzi LabView 2012 SP1. Při instalaci LabView se zpravidla automaticky nainstaluje i NI MAX. Tento program slouží jako správce zařízení, v němž lze vidět všechna zařízení, jejich konfigurace, lze tato zařízení konfigurovat apod. Kvůli neznámé chybě ve verzi 5.4 jsem byl nucen stáhnout aktuální verzi programu NI MAX 5.5. Pomocí tohoto programu lze také instalovat programy či drivery na zařízení (o čemž se zmíním v další podkapitole). Dále jsem nainstaloval FPGA a Real-time moduly společně s drivery pro CompactRIO a kompilační sadou nástrojů (využil jsem té od Xilinx), aby bylo možno nahrávat programy do FPGA.

Se softwarem se objevily nemalé potíže, několikrát bylo potřeba přeinstalovat veškerý software od National Instruments kvůli náhodně vzniklým chybám nebo náhlé nefunkčnosti různých částí těchto nástrojů. Nakonec se však vše podařilo nainstalovat správně.

3.1.3 Konfigurace a zprovoznění

Když je nainstalován veškerý software, můžeme připojit zařízení. Zapojíme všechna zařízení podle hlavního schématu na obrázku 19. Napájení zapneme nejprve pro CompactRIO, abychom zbytečně nenapájeli motor, když ještě není ověřena funkčnost komunikace. Propojíme počítač s CompactRIO (já jsem použil ethernetovou přípojku). Poté je třeba staticky nastavit IP adresu připojení (v OS Windows 7 v Centru sítí a sdílení >> Připojit k místní síti >> Vlastnosti >> Vlastnosti Protokolu TCP/IPv4 >> 169.254.84.1, maska podsítě 255.255.0.0). Tímto se obě zařízení komunikačně spojí. Nyní spustíme nainstalovaný program, NI MAX. Po jeho spuštění se spustí automatická detekce zařízení a v případě správného spojení se zobrazí zařízení NI9076 – to je zařízení CompactRIO. V MAXu můžeme zařízení resetovat, pracovat se softwarem, který je nainstalovaný na zařízení CompactRIO (při práci se software na CompactRIO je potřeba CompactRIO přepnout do nouzového režimu). Je zde dobré (spíš nutné)

nainstalovat drivery do samotného CompactRIO. Je také možné zařízení zabezpečit heslem za účelem ochrany duševního vlastnictví v podobě zdrojových kódů vytvořených programů. Nakonec je zde i možnost zařízení přejmenovat. Po rozkliknutí zařízení NI 9076 lze vidět také všechny moduly připojené k šasi CompactRIO (v našem případě modul NI 9501). Nyní budeme moci přejít k samotnému programování.



Obrázek 20 - Úvodní obrazovka programu LabView, Zdroj: autor

Spustíme LabView a založíme nový projekt (*.lvproj*). V projektu je poté potřeba v záložce *My Computer* přidat nové zařízení (New >> Targets and Devices). V následujícím okně v záložce Real-Time RIO již bude vidět zařízení NI 9076, které se přidá do projektu. Zařízení lze provozovat ve dvou režimech, FPGA Interface a Real-Time Scan engine, já jsem zvolil FPGA Interface. Nyní již můžeme začít vytvářet aplikaci.

Se systémem CompactRIO je často používaná architektura programu pro řízení pohonů v reálném čase právě za pomoci FPGA Interface. Na hradlovém poli je realizována nejdůležitější část kódu, samotné zadávání povelů modulu k řízení měniče, předávání parametrů řízení modulu a podobně. Lze tedy říci, že v FPGA se realizuje komunikace s řídícím modulem NI 9501. Nové VI na FPGA vytvoříme tak, že v projektu

rozbalíme záložku *Chassi*s a klikneme pravým tlačítkem myši na FPGA (New >> VI). V rámci tohoto VI lze přidávat bloky ze záložky FPGA Interface, které komunikují s hradlovým polem.

Dále je potřeba vytvořit VI, které bude řídit běh programu. V něm realizujeme například stavový automat, který definuje stavy zařízení. Také se zde předávají vstupní a výstupní data do/z FPGA. Toto VI je vytvořeno na procesoru v CompactRIO, přidáme ho poklepáním pravým tlačítkem myši v projektu na zařízení NI 9076 (New >> VI).

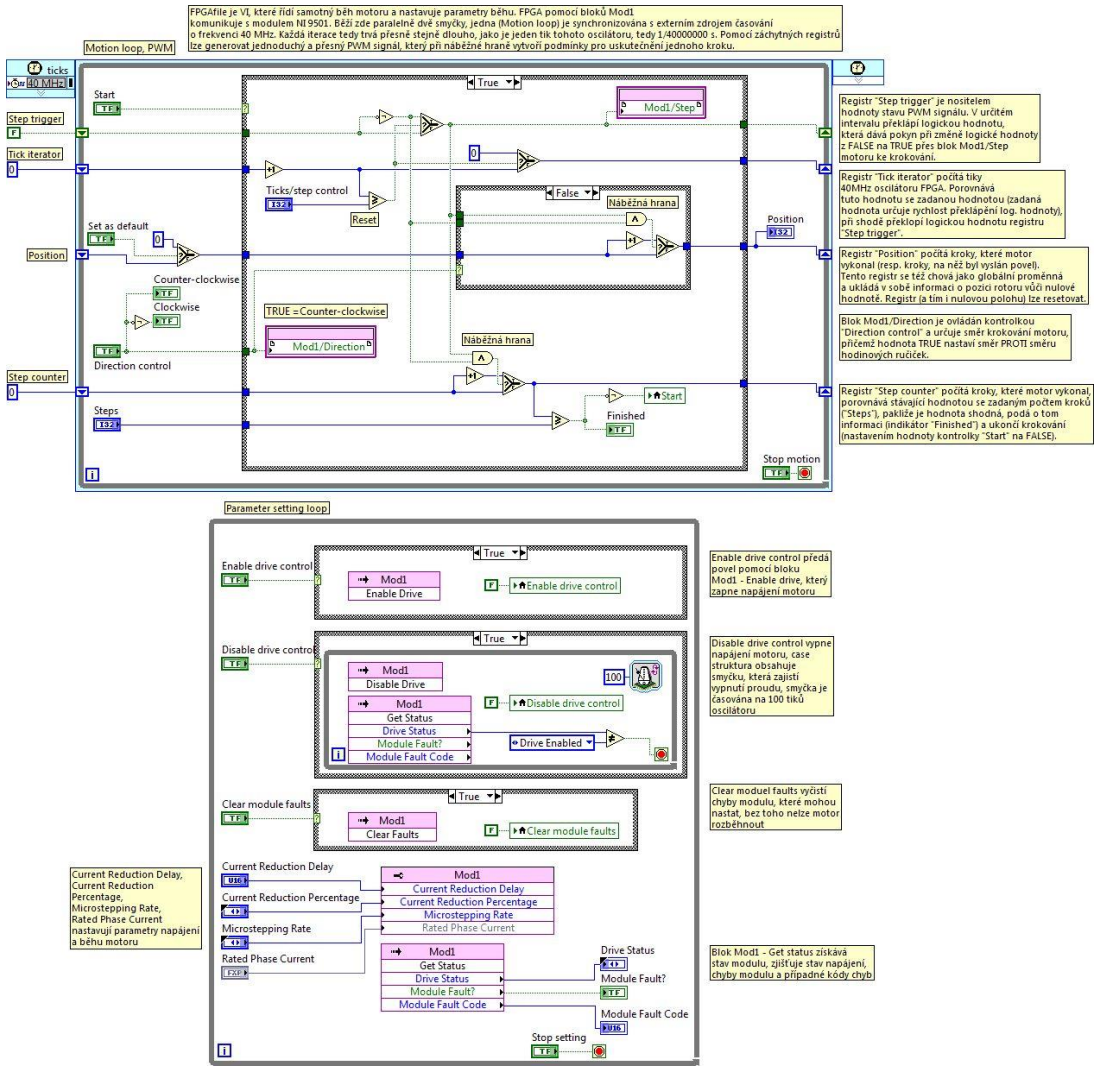
Jako poslední se vytváří VI v počítači, které slouží ke komunikaci s uživatelem. Zde jsou zapisovány vstupy od uživatele a čteny výstupy. Celá aplikace tedy funguje tak, že uživatel zadá parametry v hlavním řídícím VI, toto VI předá data pomocí sdílených proměnných do VI v procesoru CompactRIO, zde se data zpracovávají a předávají do VI v FPGA. VI v FPGA poté na základě dat od uživatele řídí měnič v modulu NI 9501 a rozbíhá krokový motor.

3.2 Vytvoření řídícího algoritmu

V této části práce popíšu řídící algoritmus, který jsem vytvořil v LabView. Zmíním se o jeho architektuře, použitých strukturách a funkčních blocích. Celý kód je sestaven z 3 souborů VI – FPGA VI, Real-time VI (tyto dva jsou nahrány v CompactRIO) a Control VI (v paměti počítače), které jsou provázané sdílenými proměnnými (sdílené proměnné jsou nutné pro zajištění přenosu dat mezi různými zařízeními).

3.2.1 FPGA VI

Nejprve se budu věnovat FPGA VI (název souboru je FPGAfile.vi), které plní zásadní roli, tedy komunikaci s hardwarem a ovládání samotného pohonu. Tato část kódu je nahrána přímo na hradlové pole FPGA, protože je důležité její přesné časování. Kvůli omezeným vlastnostem FPGA je omezen počet funkcí, které lze použít při programování kódu.



Obrázek 21 - Ukázka algoritmu realizovaného na FPGA, Zdroj: autor

Na obrázku 21 je ukázka algoritmu nahraného v FPGA, který byl vytvořen k ovládání motoru. Algoritmus je sestaven z 2 cyklů, jeden je obyčejná *While* smyčka, druhý je tzv. *Single-cycle timed loop*. Oba tyto cykly běží paralelně, což je umožněno právě realizací v FPGA. Kdybychom ten samý kód chtěli programovat pro procesor, bylo by zapotřebí vícejádrového procesoru.

Dolní cyklus je *While* smyčka. Tento cyklus slouží k zapnutí a vypnutí napájení motoru, k získávání informací o modulu a k nastavování parametrů běhu motoru. Mezi volitelné parametry běhu motoru patří Měrný fázový proud (*Rated phase current*), Procentuální snížení proudu (*Current reduction percentage*), Zpoždění snížení proudu (*Current reduction delay*) a úroveň Mikrokrokování (*Microstepping rate*). Tyto

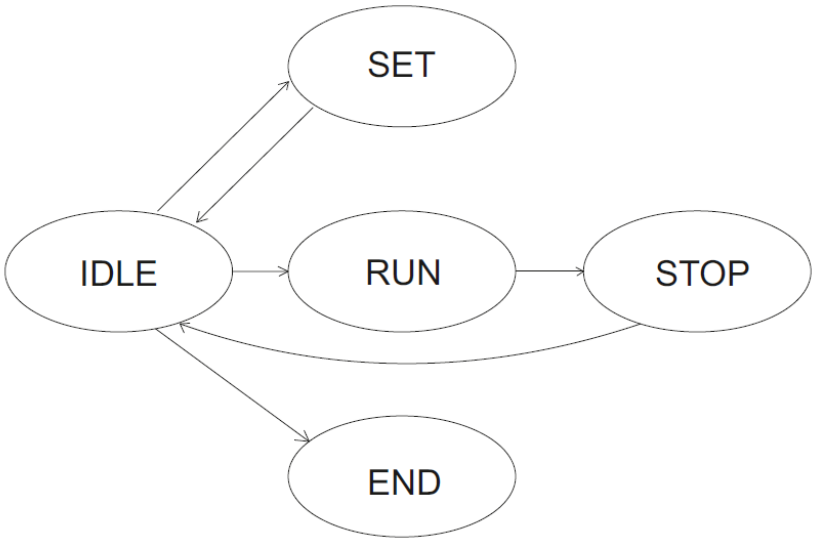
parametry lze nastavit pomocí metody *Mod1*. Dále jsou v tomto cyklu získávány informace o modulu – stav napájení (*Drive status*), Chyba modulu (*Module fault*) a Kód chyby (*Module fault code*), dále i metoda pro vyčistění všech chyb modulu (*Clear all faults*). Důležitou částí jsou zde metody *Enable drive* a *Disable drive*, které ovládají napájení motoru. Všechny tyto metody dostávají instrukce od kontrolek (např. *Enable drive ctrl*), které jsou ovládány z Real-time VI (v následující podkapitole), indikátory (např. *Drive status*) zase posílají data do Real-time VI ke zpracování a zobrazení.

Horní cyklus je přesně časovaný cyklus, synchronizovaný s externím zdrojem synchronizace FPGA o frekvenci 40 MHz takovým způsobem, že jedna iterace cyklu proběhne přesně za dobu jednoho tiknutí oscilátoru (proto název *Single-cycle timed loop*). Máme tedy velmi přesný časovací nástroj. V tomto cyklu je realizován PWM signál, který uskutečňuje samotný běh motoru pomocí metody *Step*. Je zde také metoda, která určuje směr krokování motoru (*Direction*). Metody jsou opět řízeny kontrolkami nebo posuvnými registry, jejichž hodnoty se mění na základě iteračních proměnných a zadávaných hodnot.

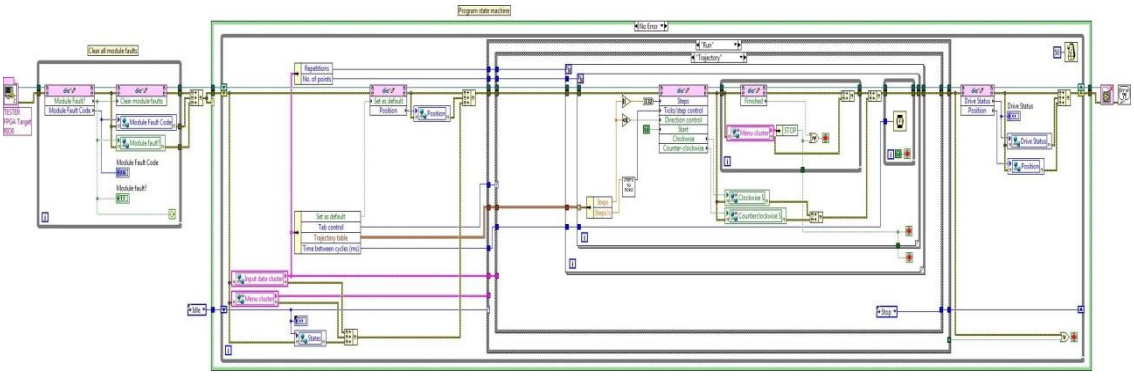
Jsou zde 4 posuvné registry. První (*Step trigger*), jenž je datového typu *BOOLEAN*, posílá pomocí metody *Step* signál modulu ke krokování při každé náběžné hraně (změně hodnoty z *FALSE* na *TRUE*). K překlápění hodnoty tohoto registru dochází na základě hodnoty dalšího registru – *Tick iterator*. Tento registr počítá iterace cyklu a při určitém počtu iterací (daném informací z kontrolky *Ticks/step control*) překlopí/zneguje hodnotu *Step trigger* registru, poté vynuluje svoji hodnotu. Tímto získáme „booleanovský“ PWM signál, jehož frekvenci lze řídit kontrolkou *Tick/step control*. S každou náběžnou hranou tedy motor udělá jeden krok (v případě, že má zapnuté napájení). Další dva registry počítají kroky motoru, jeden z nich si pamatuje polohu vůči výchozí pozici. Druhý (*Step counter*) počítá kroky a v případě, že bylo dosaženo určitého počtu kroků (daného hodnotou kontrolky *Steps*), uloží do indikátoru *Finished* hodnotu *TRUE* a změní hodnotu kontrolky *Start* na *FALSE*, čímž de facto zkratuje krokovací algoritmus a ukončí tak krokování. Díky tomu lze krokovat o přesně daný počet kroků.

3.2.2 Real-time VI

Další na řadě je Real-time VI. V tomto souboru je realizován algoritmus běhu řídícího programu v podobě stavového automatu. Na základě instrukcí uživatele je přepínán stavový automat do různých stavů, v nichž jsou prováděny určené operace (nastavování hodnot, běh motoru, ukončení programu). Jedná se o poměrně složitou strukturu cyklů a *Case struktur*, detailnější obrázky jsou dodány v příloze.

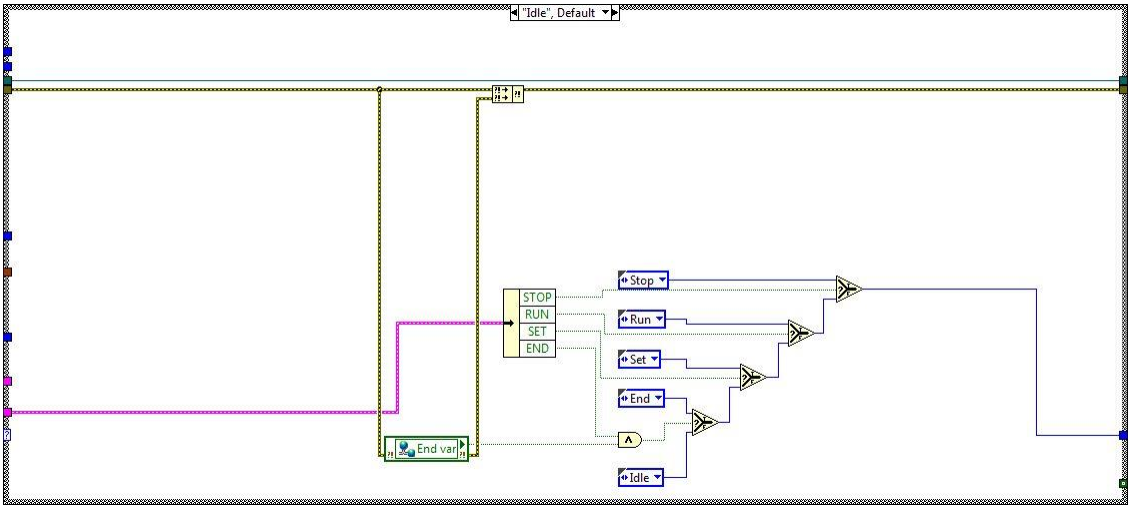


Obrázek 22 - Schéma stavového automatu, Zdroj: autor



Obrázek 23 - Ukázka algoritmu Real-time VI a stavového automatu, Zdroj: autor

První věcí po spuštění tohoto kódu je otevření reference na FPGAfile.vi, tedy VI, které je nahráno v FPGA. Tato reference je na konci běhu programu opět uzavřena. Poté následuje běh smyčky, která zachytává chyby modulu NI 9501 a případné chyby čistí, výsledky zobrazí na indikátorech a uloží do sdílených proměnných. Bez této filtrace chyb by modul nespolupracoval a nebylo by možné ani spustit motor.



Obrázek 24 – Ukázka části kódu zachycení instrukce uživatele k přechodu do nového stavu – zobrazený stav Idle, Zdroj: autor

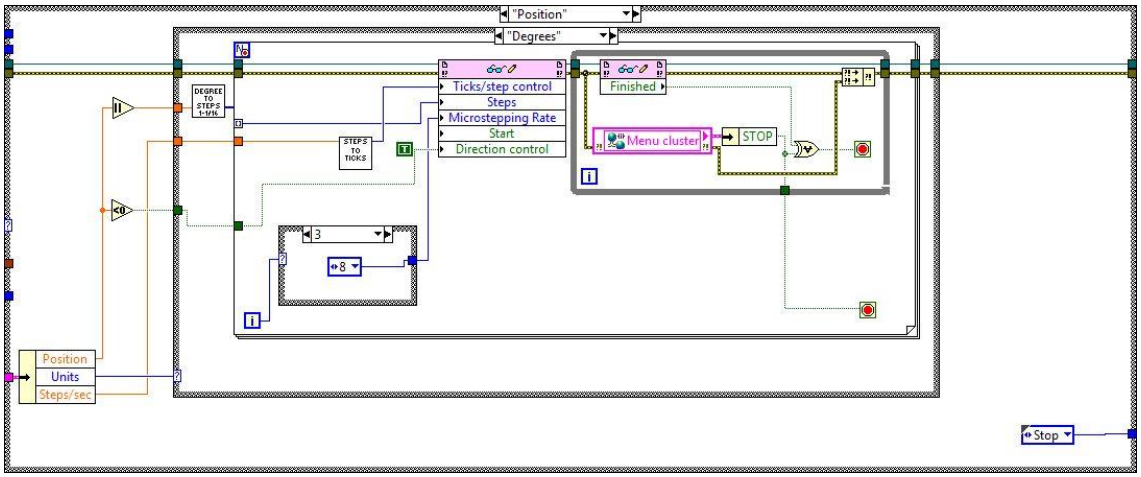
Poté již následuje samotný stavový automat. Nejprve je otestováno, zda nedošlo k chybě při otevírání reference na FPGA soubor, poté již skočí stavový automat do výchozího stavu, do stavu *Idle*. V tomto stavu čeká automat na instrukci uživatele, resp. smyčka stále běží, jen se nic neděje. V případě, že uživatel stiskne jedno z tlačítek hlavního ovládacího panelu, do posuvného registru se zapíše hodnota nového stavu a v další iteraci smyčky automat přejde do nového stavu

Na obrázku 22 je vidět schéma stavového automatu. Je zde výchozí stav *Idle*, z něhož lze přejít do stavu *Set*, který aktualizuje hodnoty nastavovaných parametrů a vrátí se zpět do stavu *Idle*. Dalším možným stavem je *Run*, který spustí krokování motoru. Po dokončení zadaných pohybů motoru nebo po stisknutí tlačítka *Stop* automat přejde do stavu *Stop*, v němž se ukončí krokování. Poté automat opět přejde do výchozího stavu.

Ve stavu *Set* jsou nastavovány parametry běhu motoru, konkrétně mikrokrokování, fázový proud, zpoždění snížení proudu a procentuální snížení proudu. Hodnoty jsou předány do FPGA pomocí uzlů. Uzly spojuje s FPGA souborem právě reference na FPGA soubor. Uzly tedy slouží k předávání dat mezi FPGA a Real-time systémem. Při prvním spuštění stavu *Set* se také zapne napájení motoru, jelikož u krokových motorů většinou vyžadujeme nenulový statický moment, tedy aby byl rotor

držen ve výchozí pozici určitým momentem. Napájení motoru se vypíná až při ukončení programu.

Ve stavu *Run* existují 3 možnosti, z nichž má uživatel na výběr, jakým způsobem chce motor provozovat. Může spustit pouhé krokování, nastavovat přesný úhel nebo počet kroků, nebo může nastavit trajektorii, po níž se motor pohybuje (určitou posloupnost pohybů o daných rychlostech a směrech).



Obrázek 25 - Ukázka kódu ve stavu Run - zaměřování polohy, Zdroj: autor

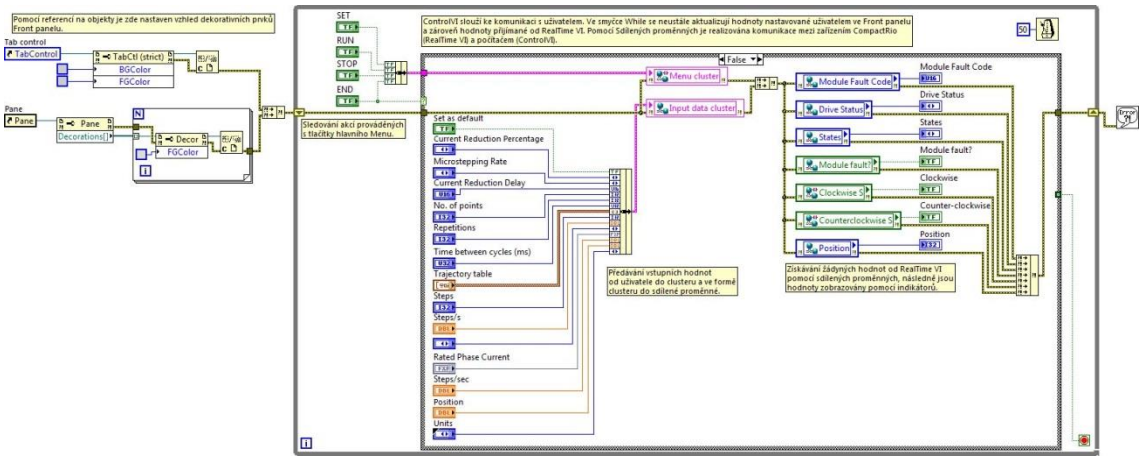
Stav *Stop* slouží k ukončení jakéhokoliv pohybu. Do FPGA pošle informaci, která zapříčiní, že se hodnota indikátoru *Finished* změní na *TRUE*, čímž v FPGA ukončí krokování a vynuluje čítače kroků (kromě registru udávajícího pozici). Poté automat skočí zpět do stavu *Idle*, v němž čeká na další instrukce.

Posledním stavem je *End*, v němž je po potvrzení vypnuto napájení motoru a ukončen program. Ovládání těchto uživatelských tlačítek je však realizováno v Control VI, z něhož jsou tyto příkazy dostávány pomocí sdílených proměnných.

3.2.3 Control VI

Control VI je jako jediné uloženo na počítači. Slouží jako uživatelské rozhraní, získává tedy hodnoty od uživatele a přes sdílené proměnné je posílá Real-time VI. Stejně tak zobrazuje hodnoty získané od Real-time VI, aby měl uživatel informace o stavu napájení, stavu automatu, směru pohybu motoru a pozici vůči výchozí poloze.

V algoritmu jsou dvě části, v první je nastavován vzhled některých dekorativních prvků (o grafickém rozhraní pojednávám v další podkapitole). Druhá část je tvořena *While* smyčkou, která periodicky sbírá data od uživatele z grafického rozhraní a předává je do sdílené proměnné. Tato smyčka také periodicky získává data od sdílených proměnných a zobrazuje je na Front panelu. V případě příkazu k ukončení programu je zde implementováno dialogové okno, v němž je potřeba potvrdit či stornovat ukončení programu.



Obrázek 26 - Ukázka kódu Control VI, Zdroj: autor

3.3 Uživatelské rozhraní

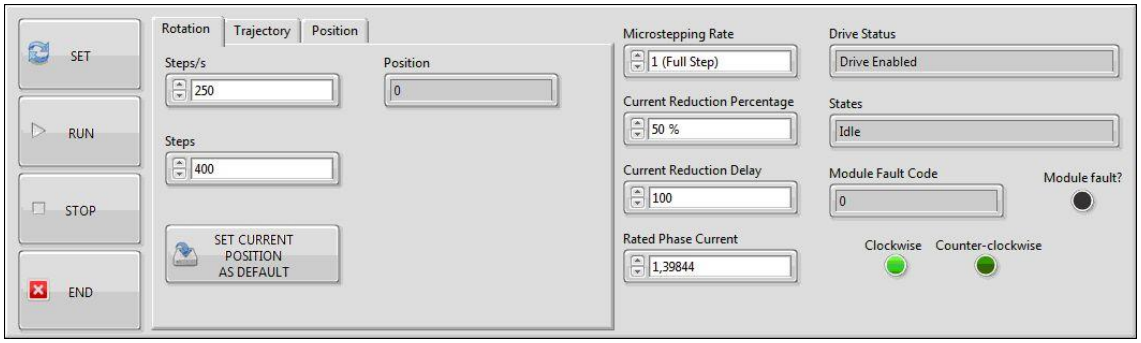
V poslední části práce se budu věnovat grafickému rozhraní. Popíšu vytvoření samotného grafického rozhraní v LabView a vysvětlím funkci programu z pohledu uživatele.

3.3.1 Tvorba GUI

Vytváření grafického rozhraní programu v LabView je poměrně jednoduchou záležitostí a je velmi intuitivní. Navíc je grafické rozhraní (Front panel) postupně vytvářeno již při tvorbě samotného kódu.

Nejprve jsem vložil do prostoru jako dekorativní prvek desku, na níž jsem poté uspořádal jednotlivé ovládací a zobrazovací prvky Front panelu. Všechny prvky jsem sladil do jednoho typu (Silver), aby měly shodný vzhled. Front panel jsem rozčlenil do 3 oblastí. Úplně vlevo je hlavní ovládací panel, kterým se přepínají stavy programu.

Uprostřed je oblast, v níž se nastavují pohybové parametry a volí se zde režimy běhu motoru. V pravé části Front panelu jsou pak indikátory, které zobrazují žádané hodnoty. Vše je lépe vidět na následujícím obrázku.



Obrázek 27 - Ukázka grafického rozhraní - Front panel, Zdroj: autor

3.3.2 Popis funkce

V této části popíšu funkci výsledného programu. Program spustíme kliknutím na bílou šipku na liště okna Front panel. Nejprve spustíme Control VI, poté můžeme spustit Real-time VI stejným způsobem. Při spuštění Real-time VI se bude program přehrávat do procesoru, takže to trvá několik sekund. Poté již máme spuštěný program.

Nyní můžeme nastavit parametry pohybu (mikrokrokování atp.), zvolíme si jeden režim běhu a nastavíme parametry pohybu (rychlost, počet kroků apod.). Stisknutím tlačítka *Set* se uloží nastavené hodnoty přes sdílené proměnné do Real-time VI a z něj přes uzly do FPGA. Zároveň se zapne napájení motoru, což bude poznat tichým syčením, motor může učinit malý krok v případě, že byl mezi polohami, tím se rotor srovnal. Na indikátorech jsou vidět údaje o motoru a modulu. Nyní můžeme stiskem tlačítka *Run* spustit námi definovaný pohyb motoru. Můžeme počkat, až motor vykoná nastavený pohyb, nebo jej ukončíme tlačítkem *Stop*. Program pak ukončíme stisknutím tlačítka *End* a potvrzením volby ve vyskakovacím dialogovém okně, nebo můžeme volbu zvrátit a program skočí opět do stavu *Idle*.

Popis funkce jednotlivých tlačítek, kontrolek a indikátorů:

SET	– aktualizuje všechny zadané parametry
RUN	– spustí pohyb motoru
STOP	– ukončí pohyb motoru
END	– ukončí program
Rotation/Trajectory/Position	– zvolí daný režim pohybu motoru
Steps/s	– nastaví rychlost pohybu v krocích za sekundu
Steps	– nastaví počet kroků
Set curr. Pos. As default	– nastaví současnou pozici jako výchozí
No. Of cycles	– nastaví počet prvků trajektorie
Repetitions	– nastaví počet opakování trajektorie
Time between cycles (ms)	– nastaví čas mezi opakovanými trajektoriemi
Units	– nastaví jednotky polohy (Degrees/Steps)
Position	– nastaví, o kolik chcete motor pootočit
Microstepping rate	– určí úroveň mikrokrokování
Current Red. Percentage	– určí procentuální snížení proudu (za účelem snížení ztrát)
Current Red. Delay	– určí zpoždění snížení proudu
Rated Phase current	– nastavuje fázový proud, defaultně je nastaven na jmenovitou hodnotu z katalogu motoru
Drive status	– indikuje stav napájení
States	– indikuje stavy stavového automatu
Module Fault Code	– indikuje kód chyby modulu
Module Fault?	– indikuje chybu modulu
Clockwise/Counter-clockwise	– indikují směr rotace motoru
Position	– indikuje polohu rotoru vůči výchozí pozici

4. Závěr

4.1 Zhodnocení

V první fázi vypracování této práce jsem se seznamoval s programovacím prostředkem LabView a se zařízením CompactRIO. Po prostudování důležitých textů a několika nezdařených pokusech se podařilo nainstalovat do osobního počítače vše potřebné (LV, drivery) pro zprovoznění pohonu.

V další fázi jsem zapojil CompactRIO a zprovoznil komunikaci mezi PC a CompactRIO s modulem NI 9501. Po resetování nastavení a vymazání software z CompactRIO jsem na něj nainstaloval drivery a zařízení se komunikačně propojila. Poté jsem k modulu NI 9501 připojil motor napájený externím zdrojem.

V poslední fázi bylo třeba naprogramovat řídící algoritmus a grafické rozhraní v LV. Postupně jsem od nejjednoduššího programu, který pouze umožňoval rotaci motoru na jednu stranu, došel k finální verzi, v níž jsou 3 režimy chodu motoru s nastavitelnými různými parametry (rychlost, počet kroků, mikrokrokování apod.).

V závěru bych rád zhodnotil několik aspektů využití systému cRIO od firmy National Instruments a softwaru LabView. Tento systém je velmi komplexní a umožňuje velmi přesné řízení. Vývojové prostředí programu LabView je velmi variabilní a dává vývojářovi téměř neomezené možnosti, jak naprogramovat zařízení a vytvořit výpočetní algoritmy.

Jeden velký nedostatek systému LabView je poměrně náročná instalace a zprovoznění softwaru. Samotná instalace může trvat i několik hodin, je potřeba nainstalovat nejprve jádro, tedy LabView, poté doinstalovat jednotlivé moduly (FPGA interface, Real-Time module atp.), navíc je potřeba instalovat všechny správné ovladače všech zařízení, která budou při realizaci projektu použita. Samotná tato instalace mi zabrala přibližně dva týdny. LabView však vidím jako jednoznačný přínos, programování v něm je jednoduché a přehledné. Díky speciálním nástrojům jsou programy a algoritmy vytvořené v LabView snadno odladitelné, testovatelné a lze je

efektivně rozšiřovat. S vytvořením řídícího programu jako takového nebyl téměř žádný problém, jediná obtížnější věc byla realizace PWM signálu na hradlovém poli, avšak i tato obstrukce byla překonána.

Systém CompactRIO má výhodu ve variabilitě, na jednom šasi lze zkombinovat několik funkcí v jednom, čímž vznikne vestavěný systém šitý na míru průmyslové aplikaci. Využití FPGA v kombinaci s mikroprocesorem na základní desce poskytuje dostatečný výpočetní výkon pro realizaci řízení pohonu. Toto zařízení se tedy jeví jako vhodný prostředek pro řízení pohonu v součinnosti s LabView.

Tato realizace řízení krokového motoru je poměrně jednoduchým příkladem, jak lze využít systém LabView a CompactRIO. Aplikaci mnou vytvořenou by jistě šlo dále zdokonalovat, přidat nové řídící prvky, přímo měřit některé veličiny pomocí dalších modulů zapojených do zařízení cRIO, což opět dokazuje snadnou práci s tímto systémem. Proto věřím, že bude v budoucnosti docházet ke stále většímu rozmachu tohoto systému ve vědě a především v průmyslu.

Seznam použité literatury

[1] **Edwards, J. D.** *Electrical Machines : An introduction to principles and characteristics.* London : Macmillan Education Ltd., 1986. 0-333-39648-0.

[2] *Freescale Technology Forum : Motor Control Part1: Trends and Roadmaps.* 2012.

[3] *Beyond Bits: Motor Control Edition.* 8. issue, 2012.

[4] Web National Instruments : Datasheet cRIO 9076. [Online] 26. 2 2014. [Citace: 3. 3. 2014.] www.ni.com. sine.ni.com/ds/app/doc/p/id/ds-354/lang/cs.

[5] National Instruments. *What is the proper programming procedure to account for timing on the NI 9501 stepper drive?* [Online] 25. 8 2010. [Citace: 27. 1 2014.] www.ni.com. digital.ni.com/public.nsf/allkb/095FC58BB279E4178625778A0056E048.

[6] Web National Instruments : Module 9501 FPGA Interface. [Online] 6 2010. [Citace: 27. 1. 2014.] www.ni.com. zone.ni.com/reference/en-XX/help/370984T-01/target4devicehelp/9501_io_reference/.

[7] Web National Instruments : Datasheet to module 9501. [Online] 8. 10 2012. [Citace: 14. 1. 2014.] www.ni.com. sine.ni.com/ds/app/doc/p/id/ds-292/lang/cs.

[8] Web Robodoupě : Web o robotice. [Online] 11. 9 2013. [Citace: 24. 4 2014.] www.robodoupe.cz. robodoupe.cz/2013/krokove-motory-1-typy-motoru/.

[9] Web National Instruments : Understanding parallel Hardware: Multiprocessors, Hyperthreading, Dual-Core, Multicore and FPGAs. [Online] 6. 12 2011. [Citace: 4. 2. 2014.] www.ni.com. ni.com/white-paper/6097/en/.

[10] Web Rascalmicro : Basic tutorial controlling motors. [Online] [Citace: 4. 2 2014.] www.rascalmicro.com. rascalmicro.com/docs-basic-tutorial-controlling- motors/.

[11] *Katalog Microcon : Kompletní pohony s krokovými motory.* 2011.

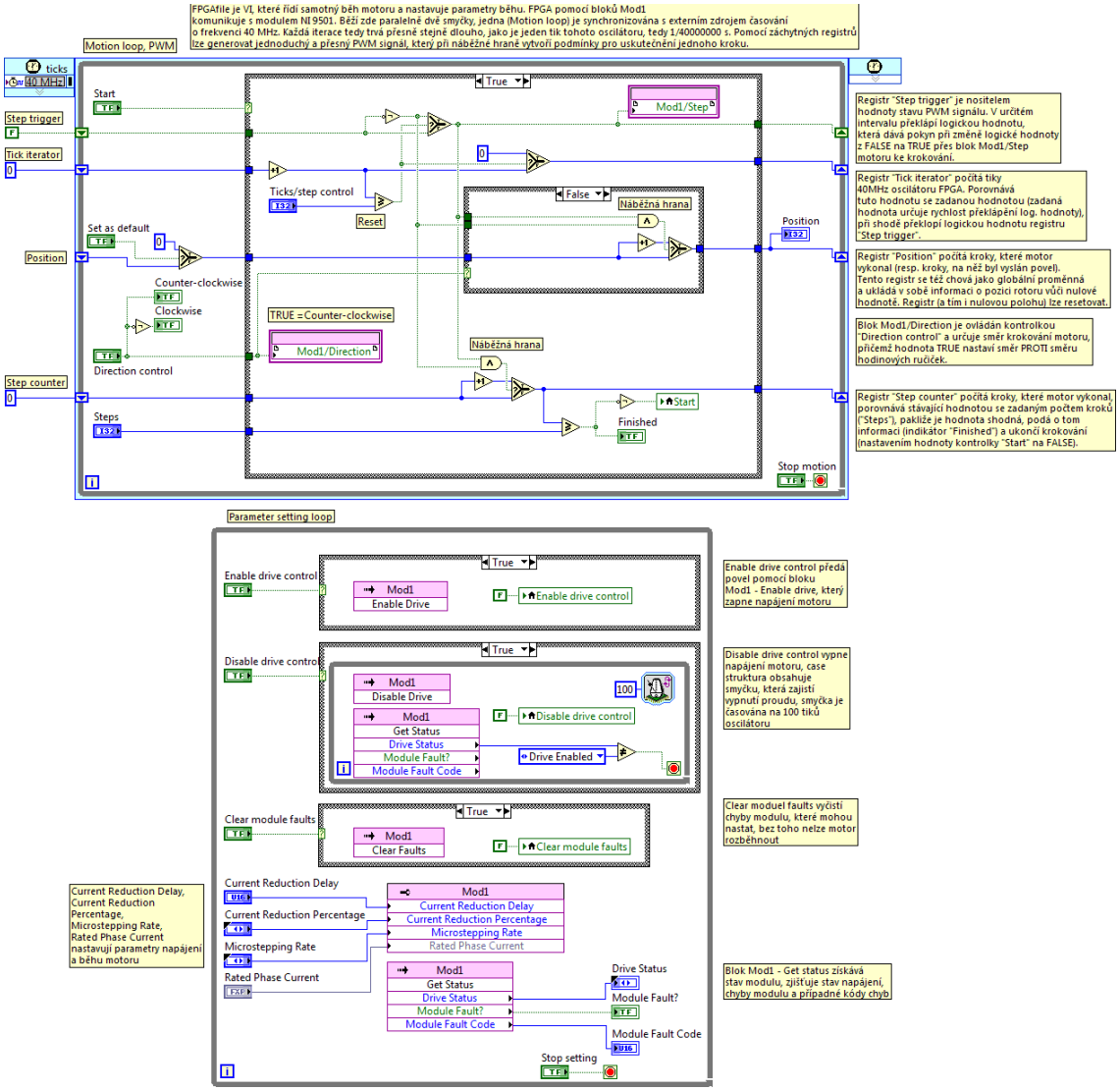
[12] **Ing. Vít Hlinovský, CSc.** *Krokové motory: podklad k přednášce předmětu A0B14AMS.* Praha, 20. 3. 2013.

Seznam obrázků

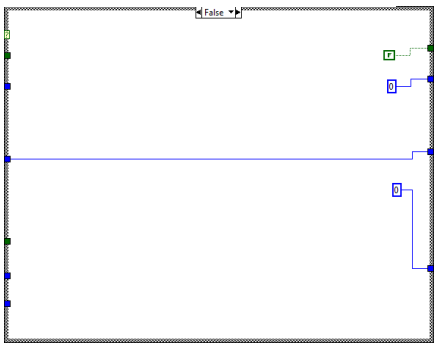
Seznam tabulek

Seznam příloh

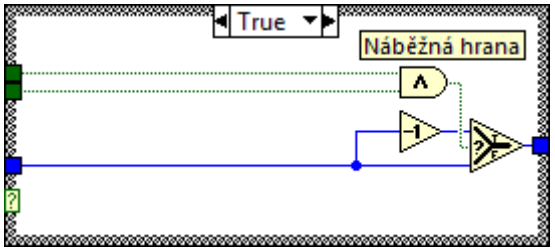
Příloha 1 - FPGA VI Block diagram



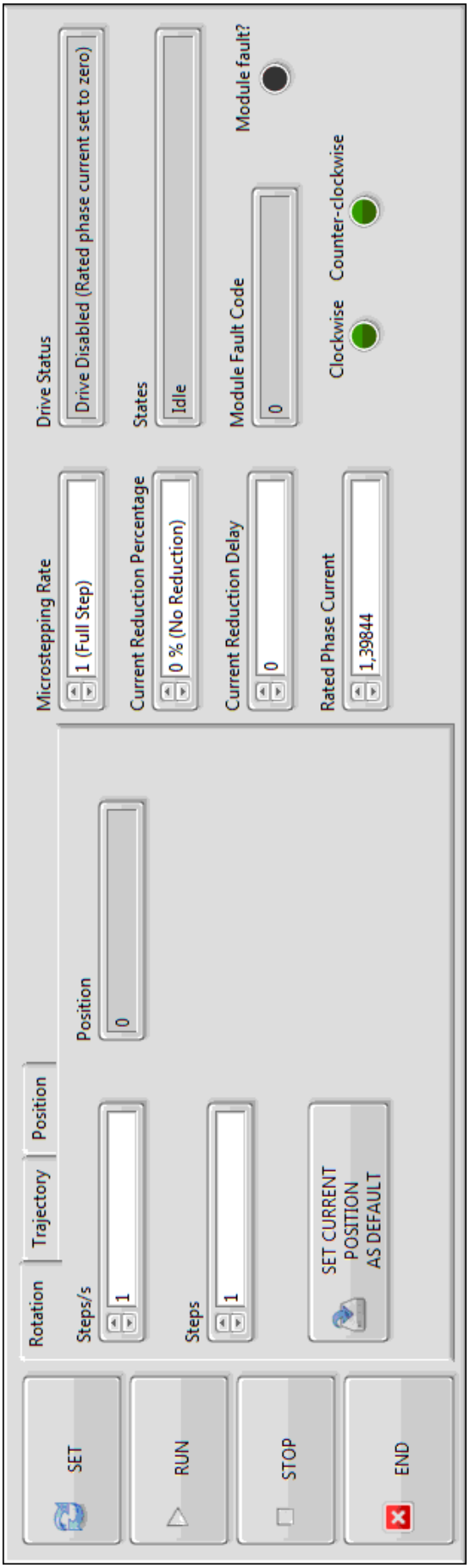
Příloha 2 - FPGA VI Block diagram Case struktura - false



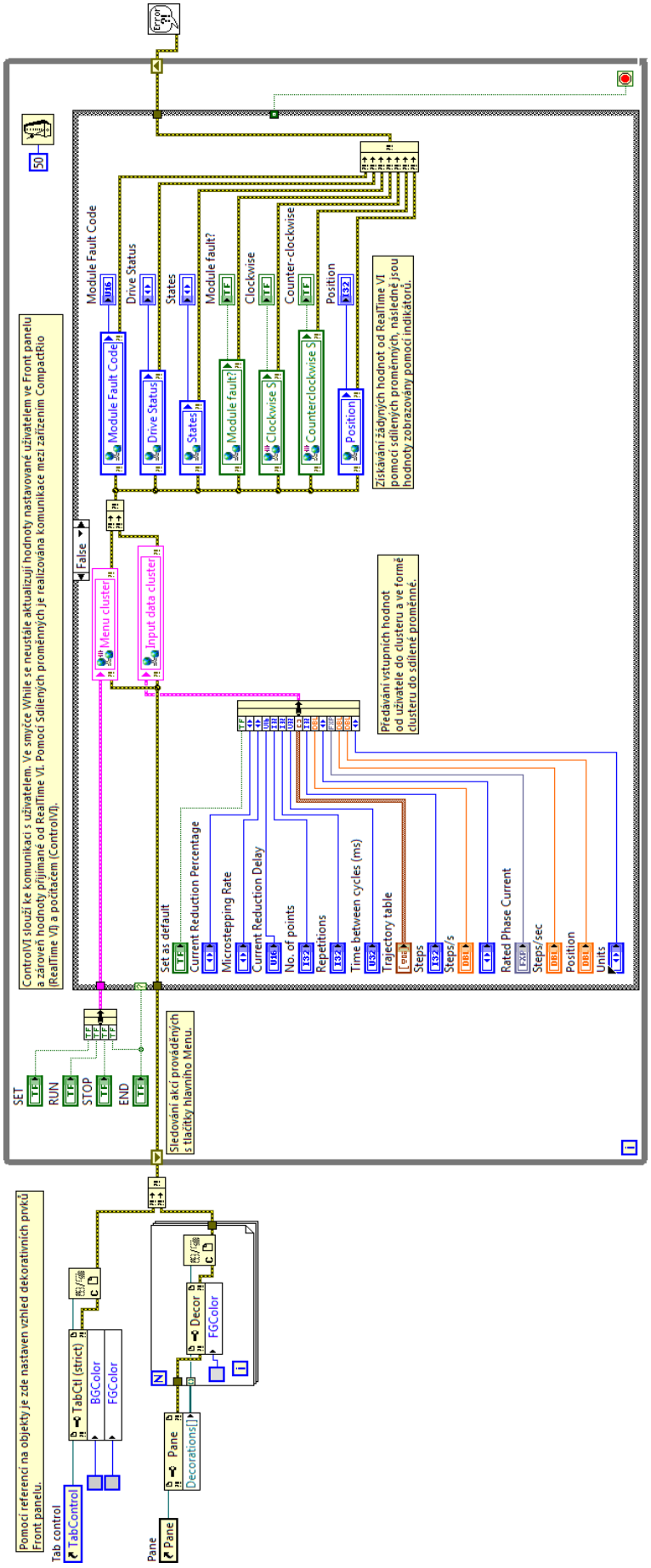
Příloha 3 - FPGA VI Block diagram case struktura - Case true - dekrementace aktuální pozice



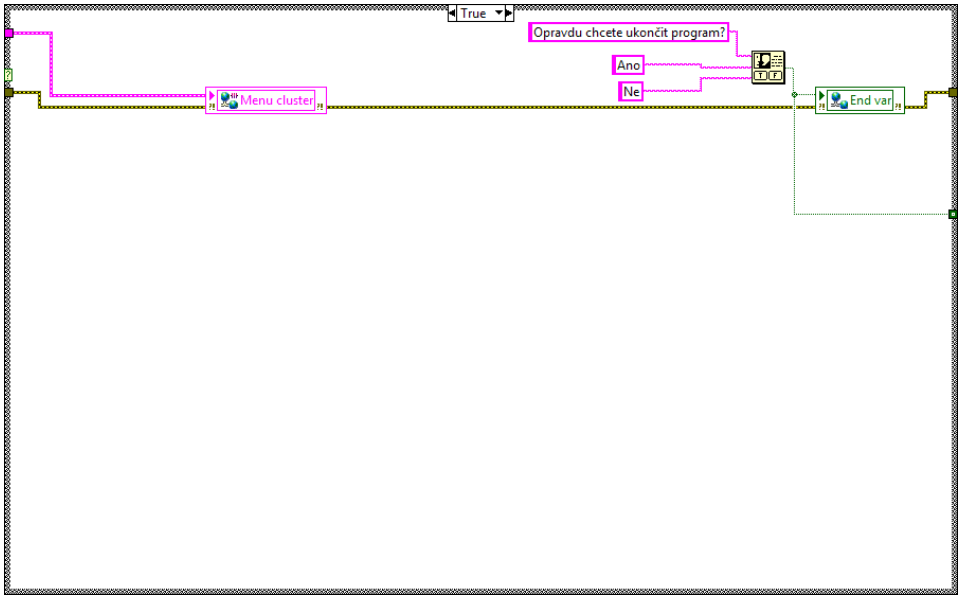
Příloha 4 - Front panel Control VI - grafické rozhraní programu



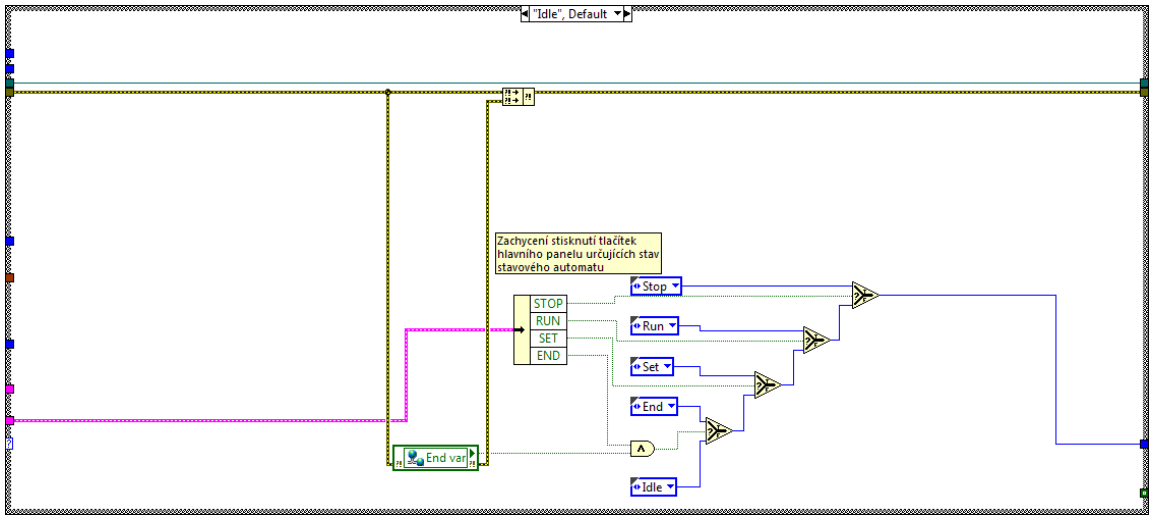
Příloha 5 - Control VI Block diagram



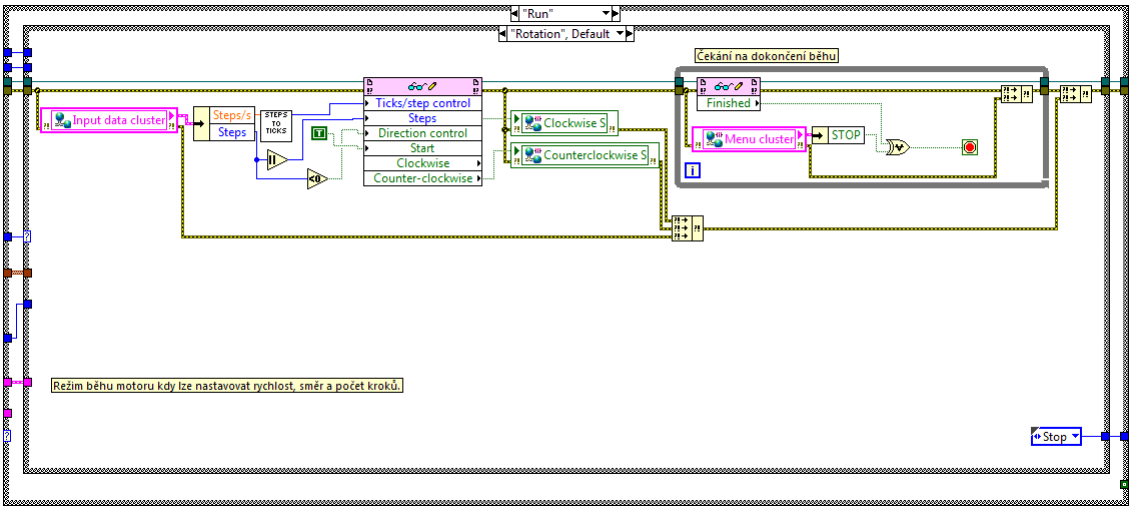
Příloha 6 - Control VI Block diagram - Case struktura - case True - ukončení programu



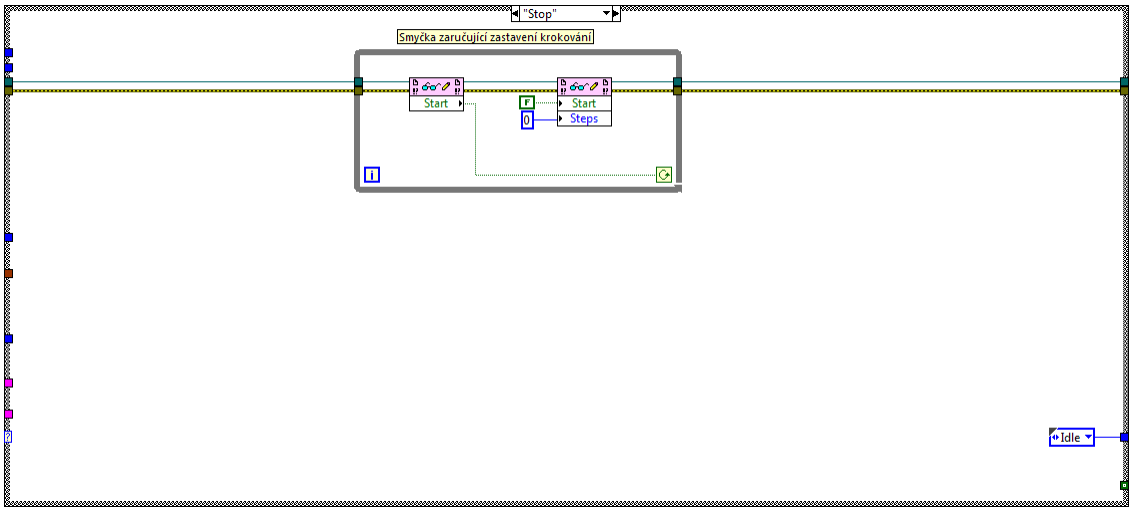
Příloha 7 - Real-time VI Block diagram - Case struktura - Stav Idle



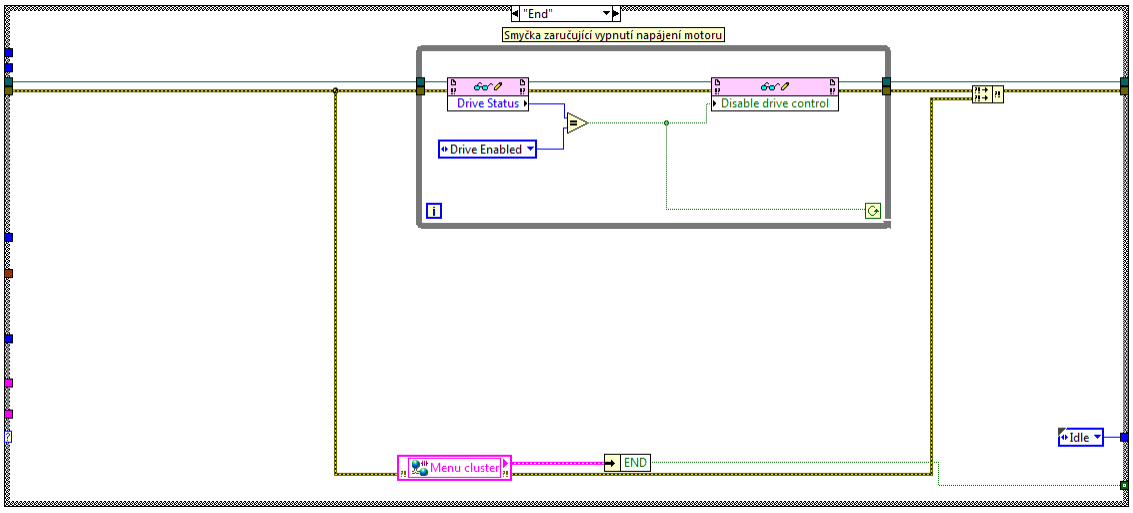
Příloha 8 - Real-time VI Block diagram - Case struktura - Stav Run



Příloha 9 - Real-time VI Block diagram - Case struktura - Stav Stop



Příloha 10 - Real-time VI Block diagram - Case struktura - Stav End



Příloha 11 - Real-time VI Block diagram