VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV VÝKONOVÉ ELEKTROTECHNIKY A ELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF POWER ELECTRICAL AND ELECTRONIC ENGINEERING

MĚŘENÍ A VYHODNOCENÍ POMOCÍ COMPACT RIO A LABVIEW

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

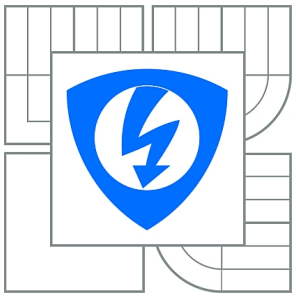
AUTOR PRÁCE
AUTHOR

PAVEL KANTOR

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV VÝKONOVÉ ELEKTROTECHNIKY A ELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF POWER ELECTRICAL AND ELECTRONIC ENGINEERING

MĚŘENÍ A VYHODNOCENÍ POMOCÍ COMPACT RIO A LABVIEW

MEASURING AND EVALUATION WITH PLATFORM COMPACTRIO AND LABVIEW

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

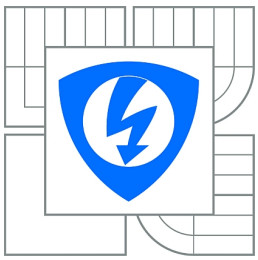
AUTOR PRÁCE — PAVEL KANTOR

AUTHOR

VEDOUCÍ PRÁCE — Ing. ROSTISLAV HUZLÍK

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav výkonové elektrotechniky a elektroniky

Bakalářská práce

bakalářský studijní obor
Silnoproudá elektrotechnika a elektroenergetika

Student:	Pavel Kantor	***ID:***	83664
Ročník:	3	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Měření a vyhodnocení pomocí Compact RIO a LabVIEW

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s programem LabVIEW a platformou CompactRIO.
2. Seznamte se s možnostmi zpracování vybraných signálů.
3. Na základě zadání vedoucího proveďte a vyhodnoťte měření.

DOPORUČENÁ LITERATURA:

Dle pokynů vedoucího

Termín zadání:	23.9.2010	***Termín odevzdání:***	30.5.2011

Vedoucí práce: Ing. Rostislav Huzlík

doc. Ing. Petr Toman, Ph.D.
Předseda oborové rady

Abstrakt

Tato bakalářská práce se zabývá návrhem a realizací měřícího programu pro měření EC motoru pomocí vývojového prostředí LabVIEW. Práce poskytuje krátký úvod do prostředí LabVIEW, ukazuje možnosti FPGA a jeho použití na platformě CompactRIO. Popisuje metodiku měření a uvádí možnosti jejího aplikování do prostředí LabVIEW. Dále se práce zabývá rozdělením typů, konstrukcí a funkcí EC motoru. Zmiňuje možné problémy, k nimž může během řešení měřícího programu dojít, od základu popisuje tvorbu programu a vysvětluje funkci všech jeho částí.

Abstract

This bachelor's thesis deals with design and realization of measuring program for measuring EC motor per development environment LabVIEW. The thesis provides short introduction to the LabVIEW environment, shows potential of FPGA and of its use on the CompactRIO platform. It describes methodology of measurement and mentions application possibilities of this platform to the LabVIEW environment. Further, the thesis deals with sorting types, construction and function of EC motor. It mentions eventual problems, which can appear during the solving of measuring program, describes from basics the creation of the program and explains all function of all program parts.

Klíčová slova

LabVIEW; CompactRIO; FPGA; řadič Real-Time; FFT; RMS; EC motor.

Keywords

LabVIEW; CompactRIO; FPGA; Real-Time controller; FFT; RMS; EC motor.

Bibliografická citace

KANTOR, P. *Měření a vyhodnocení pomocí Compact RIO a LabVIEW*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 41 s. Vedoucí bakalářské práce Ing. Rostislav Huzlík.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Měření a vyhodnocení pomocí Compact RIO a LabVIEW jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne …… Podpis autora ……..

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Rostislavu Huzlíkovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne …… Podpis autora ……..

Obsah

SEZNAM OBRÁZKŮ

SEZNAM SYMBOLŮ A ZKRATEK

Zkratka	Název	Jednotka
I_{RMS}	Efektivní hodnota střídavého proudu	A
U_{RMS}	Efektivní hodnota střídavého napětí	V
R	Elektrický odpor	Ω
T	Perioda	s
W	Energie přeměněná v teplo	J
p(t)	Okamžitý výkon	W
i(t)	Okamžitý proud	A
F(ω)	Spektrum řady konečné délky	

CompactRIO	Compact Reconfigurable Input Output
LabVIEW	Laboratory Virtual Instrumentation Engineering Workbench
EC	Electronical comutated
VI, Vis	Virtual Instrument(s)
FPGA	Field Programmable Gate Array
subVi	sub Virtual Instrument
VLSI	Very Large Scale Integration
SRAM	Static Random Acces memory
EEPROM	Electrically Erasable Programmable Read Only Memory
ASIC	Application Specific Integration Circuit
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
RMS	Root Mean Square
BLDC	Brush Less Direct Current
FIFO	First In First Out

Úvod

Jedním ze směrů, kam se současný trend vývoje řídících a měřících systémů ubírá, je možnost zpřístupnění programování těchto systémů co možná nejširšímu poli uživatelů. Snahou mnoha firem je tedy vývoj uživatelských programů zjednodušit. Jednou takovou firmou je společnost National Instrument, která přináší své vlastní programovací vývojové prostředí LabVIEW a platformu CompactRIO.

Hlavním cílem této bakalářské práce je vytvořit program pro měření a zpracování hodnot napětí, proudu, momentu a otáček EC motoru.

První kapitola mé bakalářské práce představí prostředí LabVIEW, vysvětlí funkci FPGA a jeho použití na platformě CompactRIO. Dále představí a popíše principy jednotlivých části platformy CompactRIO. Společnost National Instrument vyvinula CompactRIO nejen jako systém řízení, ale i jako měřící systém. Díky tomu má platforma ještě větší uplatnění a je s oblibou používána mnoha uživateli.

V další kapitole je vzhledem k cíli práce potřeba popsat metodiku měření a uvést možnosti její aplikace do prostředí LabVIEW.

Následující kapitola se zabývá teorií EC motoru, jeho vlastnostmi, rozdělením dle typů, konstrukcí a činnostmi.

Řešení tvorby programu pro měření a programu pro zpracování výsledků je věnována poslední kapitola práce.

1 TECHNICKÉ VYBAVENÍ

1.1 Vývojové prostředí LabVIEW

Většina moderních vývojových prostředí pro programovací jazyky je typu RAD. Tato zkratka označuje Rapid Application Development a napovídá, že v daném vývojovém prostředí lze programy vyvíjet obzvláště rychle a efektivně. Zmíněná efektivnost spočívá zejména ve snadném navrhování uživatelského rozhraní programu. Stejně jako např. Delphi je vývojovým prostředím nad jazykem Object Pascal, lze také LabVIEW chápat jako vývojové prostředí nad jazykem G. Na rozdíl od Object Pascalu, kde zdrojovým kódem programu je text, zdrojovým kódem programu v jazyce G je obrázek. G tedy značí grafický programovací jazyk. Grafické programování je v současnosti technika již běžná a hojně používaná. Byla patentovaná společností National Instruments v roce 1990. Stejně jako pro textové programovací jazyky i pro jazyk G existuje kompilátor, který produkuje samostatné spustitelné programy. Tvůrci LabVIEW prohlašují, že programy vytvořené v jazyce G běží po přeložení srovnatelně rychle, jako programy napsané v jazyce C, který je obecně považován za velmi efektivní. V jazyce G má programátor k dispozici jak rychlé funkce nízké úrovně, tak i hotové komplikované podprogramy pro matematickou analýzu, statistiku, komunikaci se standardizovanými periferiemi a podobně.

LabVIEW je systém určený pro obecné programovací účely, ale navíc obsahuje také knihovny funkcí a vývojové nástroje navržené speciálně pro získávání dat a ovládání přístrojů. Programy tvořené v LabVIEW jsou nazývány Virtual Instruments (zkratka VIs; virtuální přístroje), jelikož jejich vzhled a činnost připomínají skutečné přístroje. Nicméně, VIs jsou podobné funkcím konvenčních programovacích jazyků. Každý VI se skládá z interaktivního uživatelského rozhraní a z blokového diagramu.

Uživatelské rozhraní je zastoupeno Předním panelem (Front panel) a obsahuje ovládací a indikační prvky jako jsou tlačítka, otočné knoflíky, grafy, textové okna a tabulky. Jejich rozmístění se provede podle požadavků při návrhu úlohy. Každý prvek předního panelu má svou proměnnou v Blokovém diagramu, zajišťující vazbu na uživatele.

Blokový diagram (Block diagram) je tvořen bloky umožňující provádět například matematické operace, měřit a upravovat průběhy signálů, vytvářet tabulky a vykreslovat grafy. Procesy jednotlivých bloků běží paralelně, program tedy neběží sekvenčně, ale výpočet je řízen tokem dat. Blok zahájí výpočet v okamžiku, kdy má data na všech vstupech, po zpracování posílá výsledky na všechny výstupy.

Abychom mohli vytvořený virtuální přístroj používat hierarchicky v dalších VIs, lze definovat ikonu a konektory s připojenými proměnnými z blokového diagramu. Přes tyto konektory se VIs připojuje do diagramu dat vyšší úrovně. Takto lze rozdělit aplikaci do sérií úloh, které pak mohou být znovu rozděleny, až se komplikovaná úloha změní v sérii podřízených úloh. Odtud se VI na nejvyšší programovací úrovni skládá ze souboru subVIs, které reprezentují funkce žádané aplikace. Odlaďování je o to jednodušší, jelikož můžete spouštět každé subVI zvlášť, nezávisle na zbytku aplikace.

1.2 FPGA

Trend vývoje integrovaných obvodů s velmi velkou hustotou integrace - VLSI (Very Large Scale Integration) je v současnosti velkou měrou ovlivněn technologií programovatelných hradlových polí - FPGA (Field Programmable Gate Array). Programovatelná hradlová pole jsou speciální číslicové integrované obvody obsahující programovatelné logické bloky a programem vytvářenou matici propojení. Slovní spojení „Field programmable“ znamená, že logické bloky mohou být naprogramovány až po výrobním procesu a tak FPGA může podle potřeby vykonávat jakoukoliv logickou funkce AND, NOT, OR a NOR nebo složitější funkci (matematické funkce, převodníky, dekodéry). Ve většině FPGA logické bloky zahrnují i paměťové obvody k uložení konfigurace.

[1] „Technologie užitá pro uložení konfigurace hradlového pole je nejvýznamnější faktor pro výběr součástky pro finální aplikaci. Principiálně rozeznáváme dva základní typy FPGA podle uložení konfigurace: FPGA s volatelní a FPGA s nevolatelní konfigurací“.

FPGA s volatilní konfigurací ukládá konfigurační informace do paměťových buněk typu SRAM. Výhoda takového uložení je ve snadné konfiguraci a rekonfiguraci i za běhu systému. Nevýhodou je nutnost externí paměti a jednoduchého řadiče k nastavení programovatelného obvodu po startu systému.

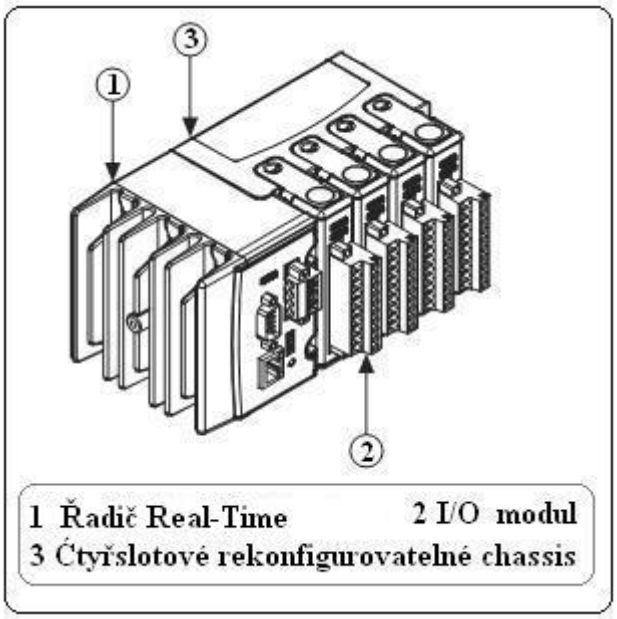
FPGA s nevolatilní konfigurací ukládá konfigurační informace obvykle do flash paměti, EEPROM, nebo do tzv. antipojistky (antifuses). Výhodou této konfigurace je nižší spotřeba a větší ochrana intelektuálního vlastnictví. Nevýhodou je obtížnější změna konfigurace, u antifuses nemožná.

Oproti ASIC (Application Specific Integration Circuit) obvodům, které jsou navrženy a vyrobeny pro specifickou úlohu konkrétního zákazníka, mají programovatelná hradlová pole řadu výhod. Patří mezi ně menší spotřeba energie, nižší výrobní náklady, kratší doba návrhu se schopností přeprogramování a opravy chyb až po výrobě, možnost opakovaného použití a schopnost měnit za běhu konfiguraci celého FPGA. Mezi nevýhody FPGA můžeme zařadit menší taktovací frekvenci a vetší složitost obvodu, který musí obsahovat paměť pro uložení konfigurace FPGA.

Programování a návrh samotných programovatelných hradlových polí není bez hlubších znalostí hardwaru vůbec snadné. Proto řada firem, ve snaze zpřístupnit FPGA co možná nejširšímu okruhu uživatelů, vyvíjí nástroje k usnadnění programovaní FPGA. Jednou z takových firem je i společnost National Instruments, která byla zmíněna již v předchozí podkapitole. Pomocí techniky RIO (Reconfigurable I/O) a vývojového prostředí LabVIEW umožňuje National Instruments uživatelům využívat výhody FPGA a zároveň jim dává volné ruce při návrhu a úpravách měřících a řídících aplikací. Uživatel v prostředí LabVIEW pomocí blokového diagramu navrhne hardware pro měření a řízení. V prostředí LabVIEW, v modulu FPGA se vygeneruje vnitřní struktura a ta se pak záhy nahraje do FPGA. Program se spouští přímo v FPGA a běží ve struktuře hradel. Příkazy programu jsou proto prováděny větší rychlostí, než by tomu bývalo třeba u PC, a všechny procesy běží na FPGA nezávisle na jiných.

1.3 CompactRIO

CompactRIO je měřící a řídicí systém s malými rozměry a odolnou konstrukcí. Jak ukazuje *Obr. 1. 1* systém je tvořen procesorem a operačním systémem reálného času (dále jen řadič Real-Time), chassis obsahující programovatelné hradlové pole (FPGA) a několik I/O modulů (vstupní/výstupní), které je možné vyměnit za chodu. CompactRIO je vestavný (embedded) systém. To znamená, že je používán také jako součást většího systému. Vestavné systémy obvykle operují v tzv. „headless" módu, tedy v módu bez uživatelského rozhraní, jako je klávesnice, monitor a myš. V tomto módu jsou aplikace často omezené množstvím paměti RAM a vestavný systém je omezen také svými rozměry.

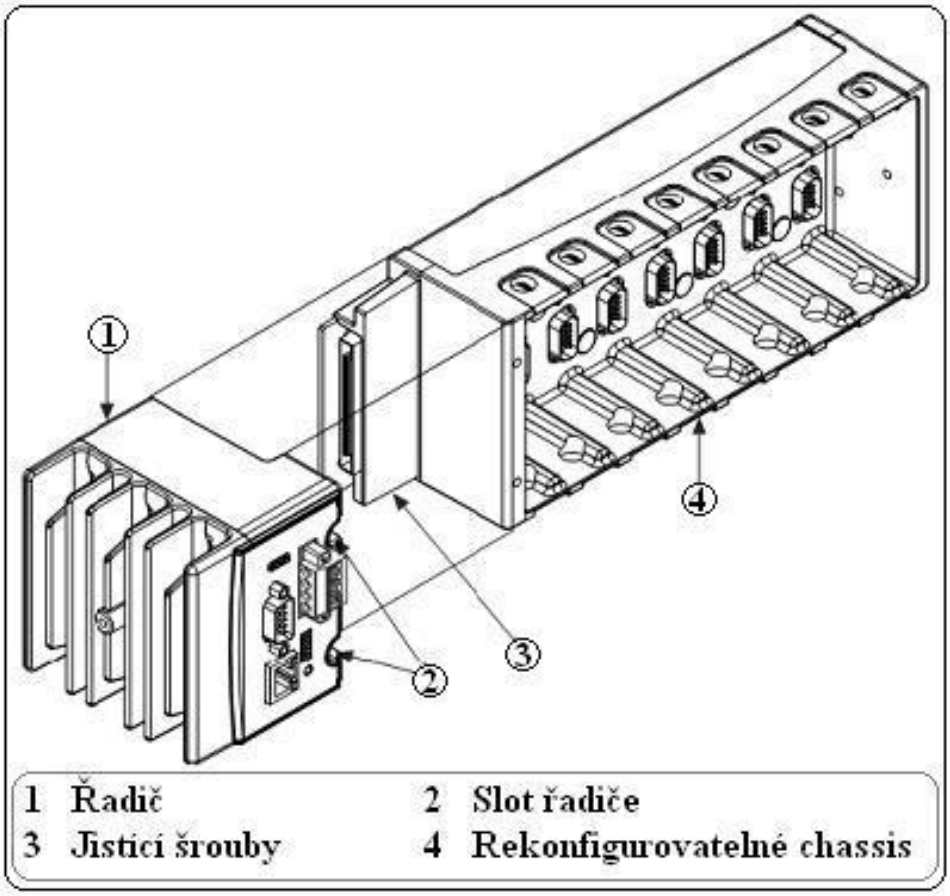


Obr. 1. 1 NI CompactRIO[2]

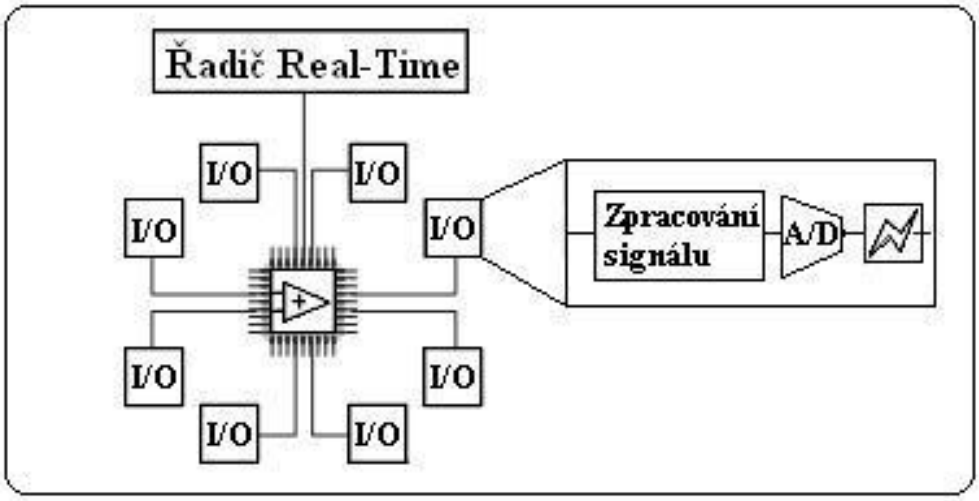
Kovové víceslotové rekonfigurovatelné **Chassis** (*Obr. 1. 2*) obsahuje jeden až tři milióny hradel FPGA (dle typu chassis), dále sloty pro řadič, I/O moduly a PCI bus rozhraní mezi FPGA a řadičem Real-Time. FPGA čip je zapojen přímo k I/O modulům, tak jak ukazuje *Obr. 1. 3*, pro přesnou kontrolu a flexibilitu v časování, spuštění a synchronizaci. Chassis má zabudovanou pomocnou datovou sběrnici k posílání dat do začleněného procesoru, který zajišťuje real-time analýzu, následné zpracování, záznam dat nebo síťovou komunikaci s připojeným hostitelským PC. Chassis a FPGA jsou přímo spojeny s obvody každého I/O modulu použitím funkcí LabVIEW FPGA I/O. Komunikace mezi I/O moduly a FPGA je digitální, neboť každý I/O modul obsahuje zabudované obvody pro úpravu signálu, A/D nebo D/A převodník a ochranu proti přepětí.

Řadič Real-Time (*Obr. 1. 4*), namontovaný v chassis, obsahuje procesor s pohyblivou čárkou (floating-point). Společnost National Instruments vyrábí několik typů řadičů s různými specifickými vlastnostmi a funkcemi, vše lze nalézt na jejich webových stránkách. Řadič používá k připojení 10/100Mb Ethernetový port (RJ-45) i sériové rozhraní USB a RS-232. Dále řadič obsahuje dvojité stejnosměrné napájecí vstupy 9-35 V, uživatelský DIP přepínač, LED indikátory stavu, hodiny reálného času a hlídací časovače (watchdog timers). V řadiči je zabudována trvalá paměť (při odpojení napájení paměti se její obsah nesmaže) pro ukládání trvalých dat. Kapacita paměti řadiče se s každým novým typem neustále zvětšuje. LabVIEW Real-Time modul používá

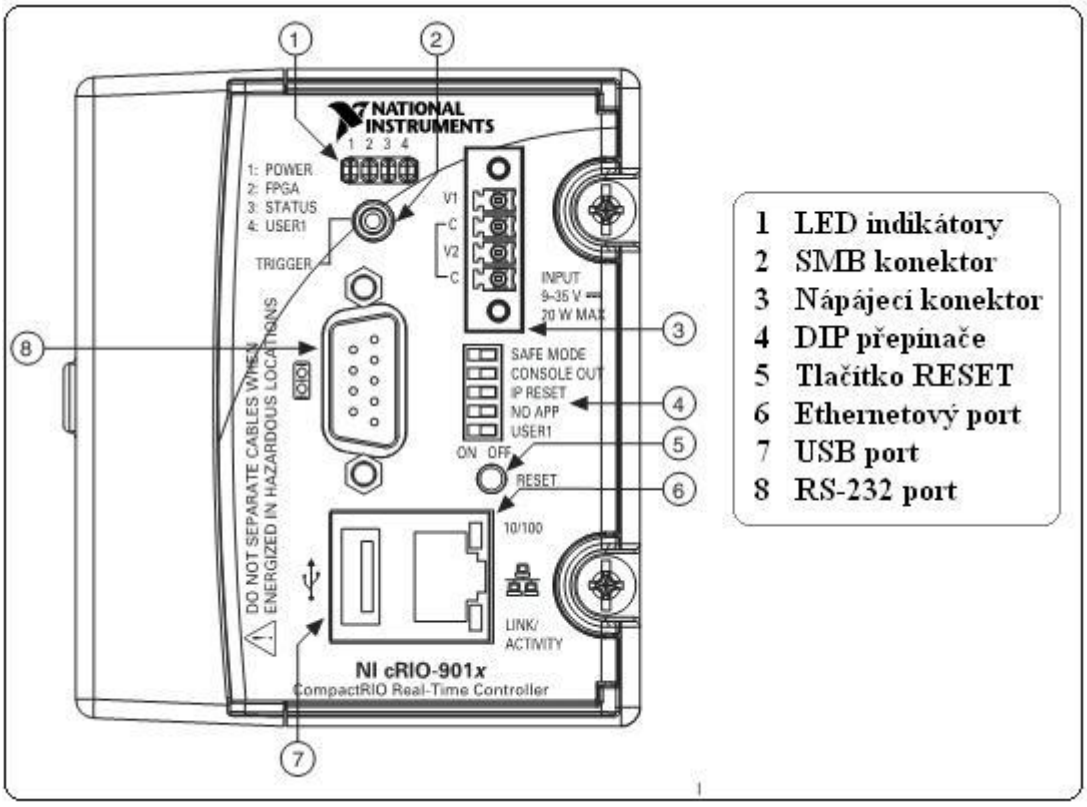
svou vlastní paměť DRAM. Protože je DRAM v porovnání s trvalou pamětí mnohem rychlejší, využije řadič paměť DRAM při běhu aplikace k ukládání proměnných programu, kopii programu, polí a části operačního systému a LabVIEW Real-Time modulu. Po vypnutí modulu se všechny informace v této paměti smažou. Některé systémy CompactRIO podporují tzv. Low-power sleep mode, ve kterém má systém minimální spotřebu. Tím se také méně zahřívá než v normálním režimu. Obvykle pak ale nelze komunikovat s moduly.



Obr. 1. 2 Rekonfigurovatelné chassis a řadič[2]

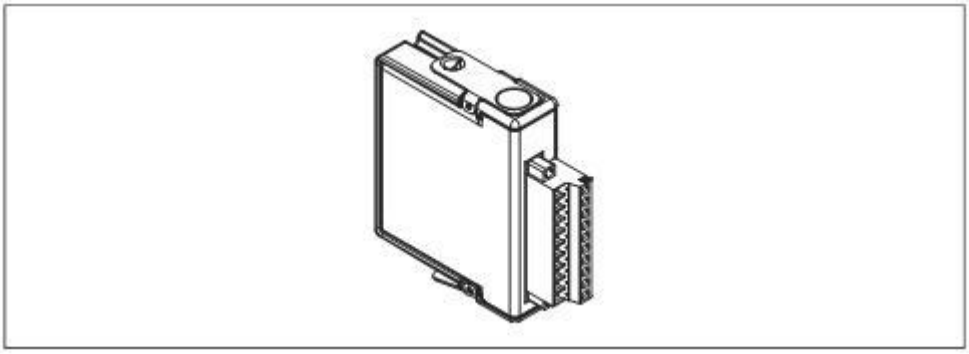


Obr. 1. 3 Topologie připojení[2]



Obr. 1. 4 Řadič Real-Time[2]

I/O moduly systému CompactRIO komunikují do a zpět z externích zařízení, jako jsou čidla a akční členy, a jsou dále přímo propojeny a komunikují s FPGA. Mezi FPGA a řadičem Real-Time jsou data přenášena přes CompactRIO PCI bus. Každý CompactRIO modul (*Obr. 1.5)* spojuje externí zařízení přes šroubovou svorku, BNC nebo DSUB konektory zabudované do modulů k redukci místa a usnadnění zapojování vodičů. Vodiče se obvykle zapojují přímo z modulů do čidel a akčních členů, protože moduly obsahují obvody pro úpravu signálů různých napěťových rozmezí a různých typů signálů. Obvody pro úpravu signálů leží mezi konektory a A/D, D/A převodníky.



Obr. 1. 5 I/O modul[2]

Většina aplikací běžící na CompactRIO používá tři oddělené procesy: PC s operačním systémem, řadič Real-Time a FPGA, jak ukazuje *Obr. 1. 6*. Real-Time řadič používá svůj vlastní LabVIEW Real-Time operační systém (VxWorks) místo operačního systému Windows. FPGA nepoužívá žádný operační systém, protože kód je vložen přímo do hardwaru (hradel). Na systému CompactRIO mohou běžet následující 4 konfigurace:

- FPGA, řadič Real-Time a PC s operačním systémem;
- samostatné FPGA VI;
- samostatné FPGA VI a VI řadiče Real-Time;
- FPGA VI a PC s hostitelským VI.

PC s hostitelským VI obvykle vykonává následující úkoly:

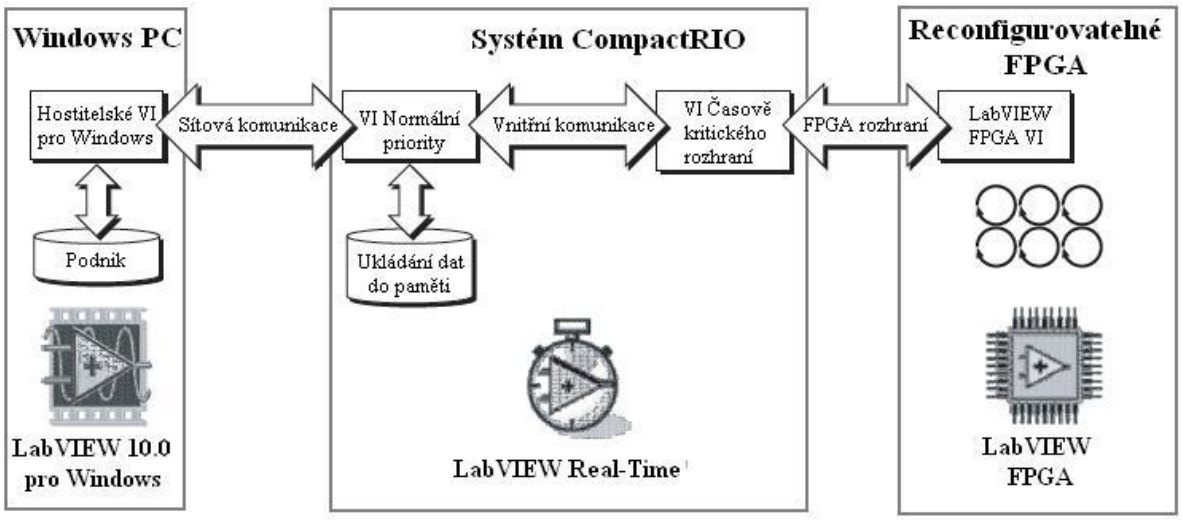
- zápis dat;
- zpřístupňování databáze;
- integrování s podnikovými systémy;
- poskytování uživatelského rozhraní.

VI řadiče Real-Time obvykle vykonává následující úkoly:

- zpracování dat;
- ovládání;
- zápis dat na řadič Real-Time.

FPGA VI obvykle vykonává následující úkoly:

- I/O (vstupy/výstupy);
- hardwarově založené časové nastavení a spouštění;
- nízko - úrovňové zpracování signálu;
- ovládání.



Obr. 1. 6 Architektura CompactRIO[2]

2 TEORIE MĚŘENÍ

2.1 RMS

Pojem RMS (z anglického root mean square), je v češtině zaveden jako efektivní hodnota. Efektivní hodnota střídavého proudu *i(t)* je definována jako hodnota stejnosměrného proudu *I*, který ve vodiči o odporu *R* za dobu *T* vyvolá stejné tepelné účinky jako proud střídavý. Vztah pro energii *W* přeměněnou v teplo v lineárním rezistoru s odporem *R* je definován jako:

$$W = R \cdot I^2 \cdot T \,. \tag{2.1}$$

Okamžitý výkon *p(t)* střídavého proudu je

$$p(t) = u(t) \cdot i(t) \tag{2.2}$$

a použitím Ohmova zákona dostaneme tvar

$$p(t) = R \cdot i^2(t) \,. \tag{2.3}$$

Celkové množství tepla, vyvinutého střídavým proudem za dobu *T*, je integrál okamžitých výkonů

$$W = \int_0^T R \cdot i^2(t)dt \,. \tag{2.4}$$

Porovnáním obou energetických účinků podle vztahů *(2.1)* a *(2.4)* dostaneme rovnici

$$R \cdot I^2 \cdot T = \int_0^T R \cdot i^2(t)dt \,, \tag{2.5}$$

odkud jednoduchými úpravami dostaneme vztah pro efektivní hodnotu proudu

$$I_{RMS} = \sqrt{\frac{1}{T}\int_0^T i^2(t)dt} \,. \tag{2.6}$$

Při odvozování efektivní hodnoty napětí vyjdeme ze vztahu

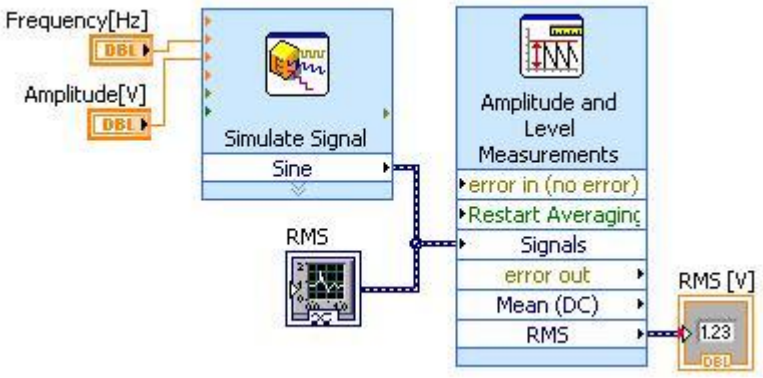
$$W = \frac{U^2}{R} \cdot T \tag{2.7}$$

a užitím obdobných vztahů jako u odvození proudu bude vzorec pro efektivní hodnotu napětí

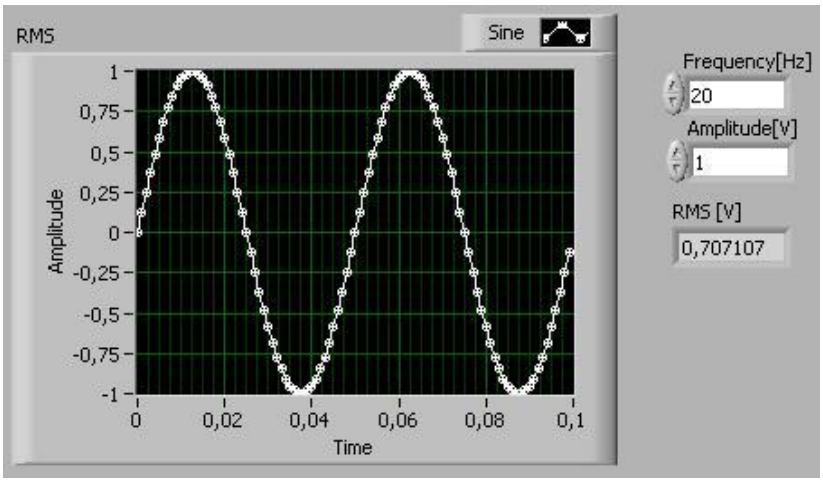
$$U_{RMS} = \sqrt{\frac{1}{T}\int_0^T u^2(t)dt} \,. \tag{2.8}$$

V této bakalářské práci je v měřícím programu ve vývojovém prostředí LabVIEW zapojeno několik bloků pro výpočet RMS hodnot. Tento blok se jmenuje Amplitude and Level Measurements a můžeme ho najít v paletě *Signal Processing >> Waveform Measurements*.

Kromě RMS hodnoty může blok spočítat i střední hodnotu a zobrazit velikosti hodnot jako je dolní špička, horní špička, špička-špička (peak to peak) a RMS 1 periody signálu. Správnou funkci bloku pro RMS si můžeme ověřit jednoduchým programem (*Obr. 2. 1)*, kde k bloku pro výpočet RMS připojíme blok k simulaci signálu Simulate Signal (paleta *Express >> Input*). Na *Obr. 2. 2* je pak vidět průběh signálu a vypočítaná hodnota RMS.



Obr. 2. 1 Blok k simulování signálu a blok pro výpočet RMS



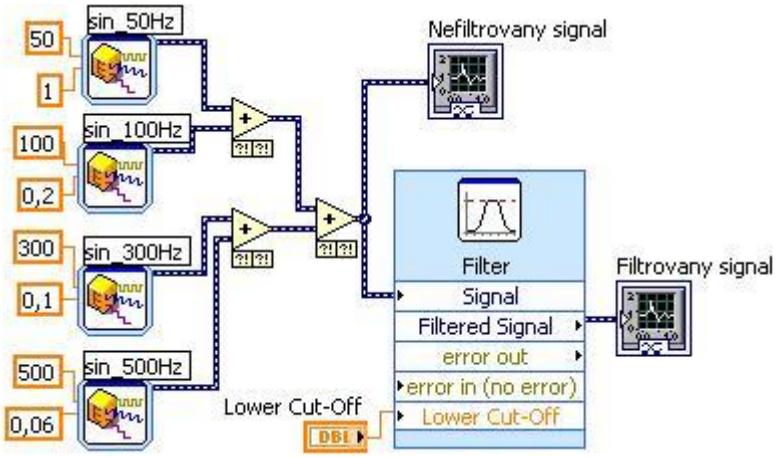
Obr. 2. 2 Průběh signálu a vypočítaná hodnota RMS

2.2 IIR Filtr

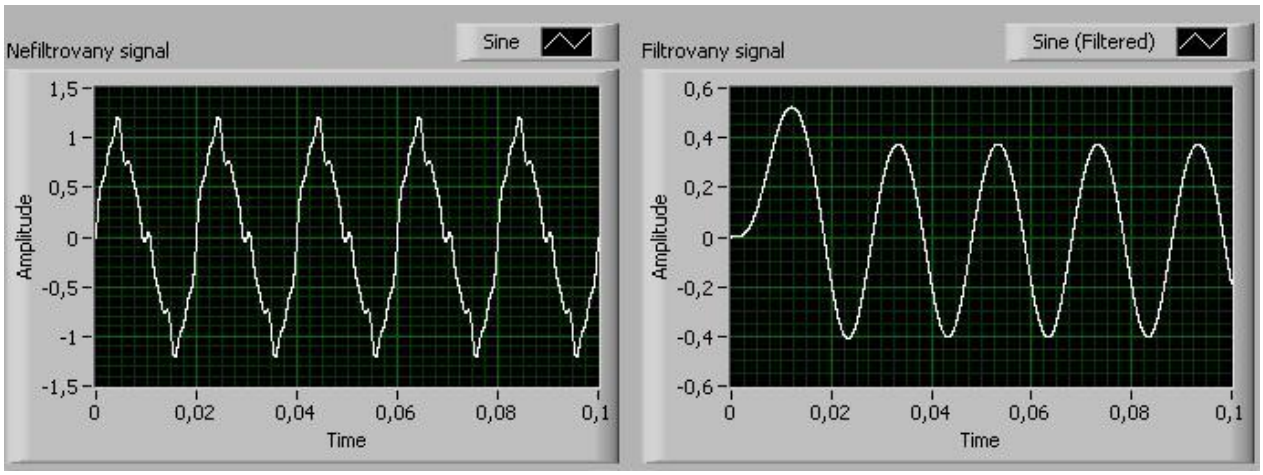
Filtry obecně slouží k úpravě signálu, ke zvýraznění nebo potlačení některých složek spektra signálu a dále ke změně jejich fázového posunutí. Spektrum signálu ukazuje, z jakých harmonických složek je signál složen, a je tvořeno informacemi o amplitudách jednotlivých složek a o jejich fázovém posunutí. Filtry můžeme rozdělit podle mnoha hledisek. V této bakalářské práci je popsán pouze filtr s nekonečnou impulzní odezvou (IIR, infinite impulse response). Je to lineární diskrétní filtr, tj. takový filtr, který vyhovuje principu superpozice. Součástí IRR filtru je nejméně jedna smyčka pro zpětnou vazbu. Tyto filtry se popisují diferenční rovnicí, která představuje algoritmus výpočtu odezvy filtru. Algoritmů je několik, mezi nejvíce používané patří Bessel, Butterworth, Chebyshev, Chebyshev2 a Eliptický algoritmus, z nichž každý je ještě určitého řádu.

V měřícím programu v LabVIEW je použito několik bloků pro filtraci signálu. Bloky najdeme v paletě *Express >> Signal Analysis* pod názvem Filter. U tohoto bloku se dá nastavit typ filtru (horní propust, dolní propust, pásmová propust, pásmová zádrž a vyhlazovací filtr) a topologie (algoritmus) výpočtu. Na *Obr. 2. 3* vidíme jednoduchý program k prezentování

funkce bloku. Pomocí čtyř bloků na simulování programu, každý o jiné amplitudě a frekvenci, vytvoříme signál o více harmonických složkách, který dále připojíme na blok filtru. Na *Obr. 2. 4* je zobrazen filtrovaný průběh signálu.



Obr. 2. 3 Bloky k simulování signálu a blok pro filtr



Obr. 2. 4 Průběh filtrovaného a nefiltrovaného signálu

2.3 DFT, FFT

Při zpracování naměřených hodnot, které jsou tvořeny vzorky signálů, budeme chtít určit spektrum signálu. K určení spektra signálu můžeme použít matematickou metodu zvanou Diskrétní Fourierova Transformace (DFT). Touto metodou se dá nejen určit spektrum signálu ze vzorků signálu, ale i signál ze vzorků spektra. Při výpočtu Diskrétní Fourierovy transformace vycházíme z Fourierovy transformace pro diskrétní signál. DFT je vymezena pro řady konečné délky. Spektrum řady konečné délky je vyjádřeno rovnicí [3]

$$F(\omega) = \sum_{k=-\infty}^{+\infty} f(k) \cdot e^{lj\omega k} = \sum_{k=1}^{N-1} f(k) \cdot e^{-j\omega k}, \qquad (2.9)$$

kde N je délka řady a f(k) = 0 pro k < 0 a pro k ≥ N, kde k = 0,1,...N-1.

Spektrum řady F(ω) je periodická funkce s periodou 2π, obsahující jednotlivé body (frekvenční složky). Tyto body se dají zapsat jako [3]

$$\omega_m = m \cdot \frac{2\pi}{N}, \qquad \text{kde m = 0,1,...N-1.} \tag{2.10}$$

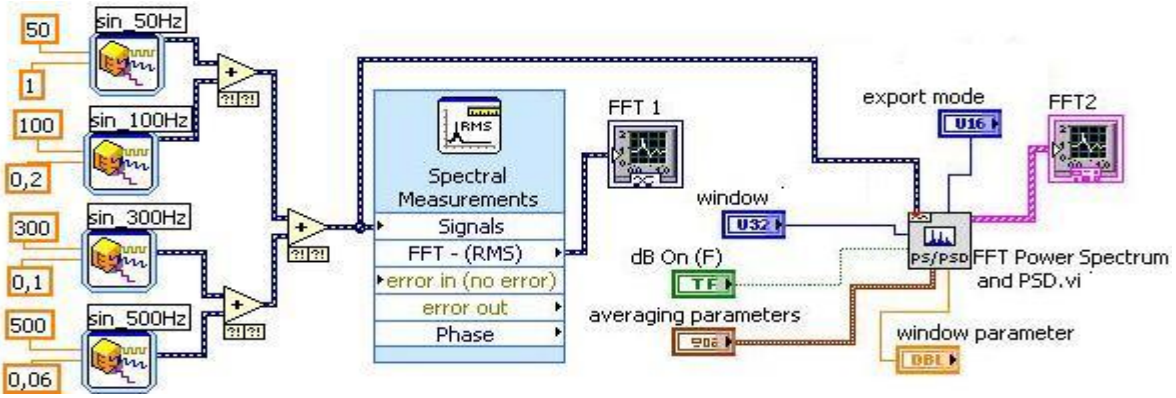
Hodnoty spektra vymezíme v diskrétních bodech rovnicí [3]

$$F(m) = F(\omega_m) = F\left(m \cdot \frac{2\pi}{N} \right), \text{ kde m = 0,1,…N-1} \tag{2.11}$$

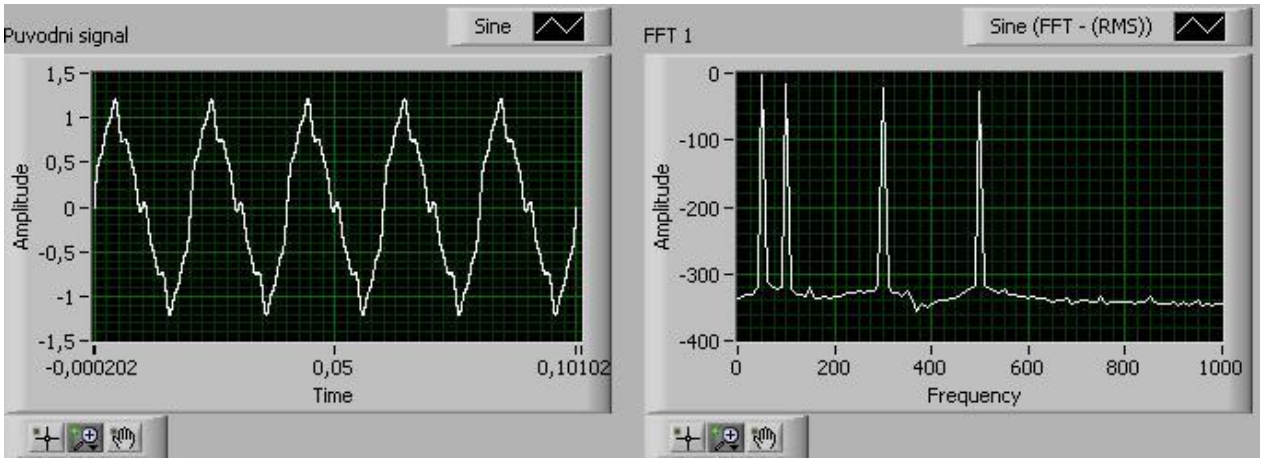
Vztah (2.10) dosadíme do vztahu (2.11) a dostaneme konečnou rovnici pro přímou diskrétní Fourierovu transformaci [3]:

$$F(m) = F\left(m \cdot \frac{2\pi}{N} \right) = \sum_{k=0}^{N-1} f(k) \cdot e^{-jmk\frac{2\pi}{N}} . \tag{2.12}$$

V roce 1965 publikovali pánové J. W. Cooley a J. W. Turkey velmi efektivní algoritmus pro výpočet DFT. Díky velké časové úspoře, kterou tento algoritmus přináší, byl pojmenován Fast Fourier Transform (FFT, rychlá Fourierova transformace). Dnes je tato matematická metoda včleněna do mnoha výpočetních programů, včetně programu LabVIEW, ve kterém ho můžeme najít tradičně ve formě bloku. Blok pro FFT s názvem Spectral Measurements (*Obr. 2. 5*) nalezneme v paletě *Programming >> Waveform >> Analog Waveform >> Waveform Measurements*. V bloku lze nastavit průměrování, nastavení fáze a okenní funkce. Ve stejné paletě jako byl předchozí blok je již předvytvořené VI s názvem FFT Power spectrum and PSD.vi (*Obr. 2. 5*). To nabízí podobná nastavení a funkce jako u bloku Spectral Measurements. Na *Obr. 2. 6* je pak původní signál a rychlá Fourierova transformace.



Obr. 2. 5 Jednoduchý měřící program s bloky pro FFT



Obr. 2. 6 Průběh původního signálu a signálu s FFT

3 EC MOTOR

V této kapitole je popsána historie, princip činnosti a konstrukce EC motoru. Zkratka EC znamená Electronical Comutated, v překladu elektronicky komutovaný. V některé literatuře se můžeme setkat se zkratkou BLDC motor (Brushless Direct Current), v překladu bezkartáčový stejnosměrný motor.

3.1 Počátky EC Motoru

Vývoj EC motoru byl podmíněn stále přibývajícími požadavky na lepší vlastnosti klasických komutátorových stejnosměrných motorů (Direct current motor). Komutátor s kartáči, který u klasického DC motoru způsobuje elektromagnetické rušení, omezuje rychlost otáčení a celkově zkracuje délku života motoru, byl nahrazen elektronickou komutací. Zároveň si EC motor uchoval dobré vlastnosti DC motorů, mezi které patří velký záběrový moment, několikanásobná přetížitelnost, nízká časová konstanta a malé rozměry.

Již v minulosti se používaly motory bez mechanických komutátorů, byly nazývány servomotory, třífázové servomotory nebo střídavé AC motory. Napájely se střídavým napětím sinusového průběhu a jejich užití bylo jen v několika speciálních aplikacích, kde byly vyžadovány konstantní otáčky (ventilátory, videokamery a videorekordéry).

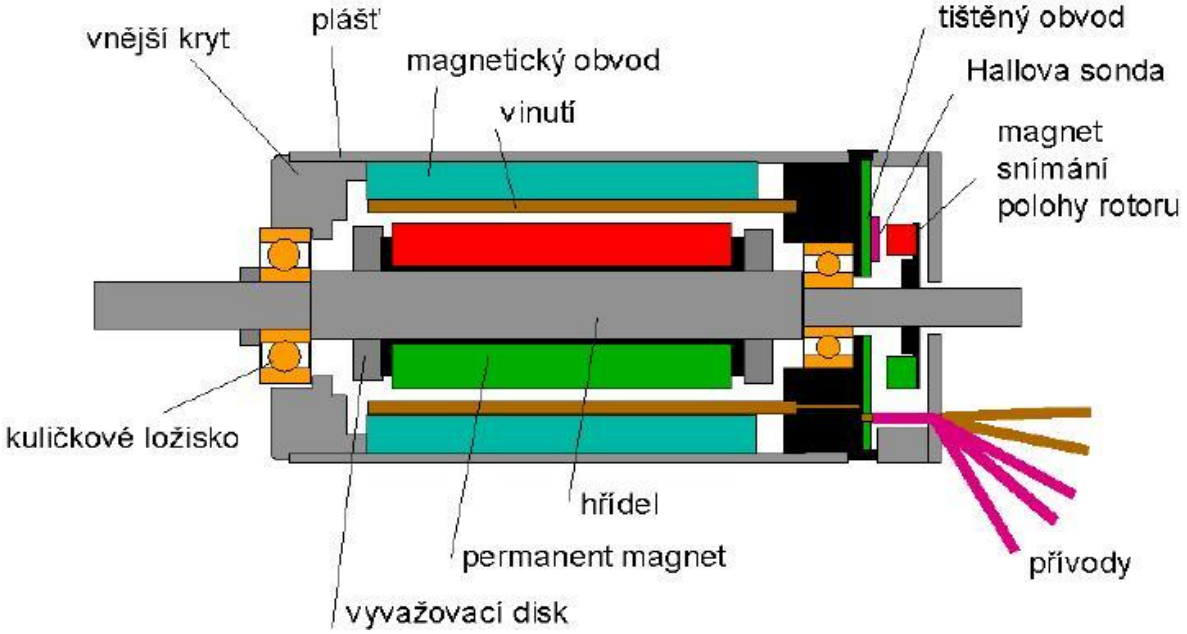
Dnes se již elektronicky komutované motory s úspěchem používají v zařízeních s vyššími otáčkami např. modelářství, kde konstrukce klasických stejnosměrných motorů s komutátorem a sběracími kartáči není příliš vhodná.

3.2 Konstrukce EC motoru

Následné rozdělení převzato z [4]
EC motory lze podle konstrukce dělit na:

- válcové EC motory;
  - s vnitřním rotorem;
    - se statorovým vinutím bez drážek;
    - se statorovým vinutím v drážkách;
    - s rotujícím jihem.
  - s vnějším rotorem.
- ploché (diskové) EC motory;
  - s vnitřním rotorem (krátký válec);
  - s vnějším rotorem (plochý diskový tvar).

Abychom u EC motoru (*Obr. 3. 1*) dosáhli stejných vlastností a charakteristik jako má DC motor, musíme udržet řízení vzájemné polohy magnetického pole statoru a magnetického pole rotoru. Pokud chceme odstranit mechanický komutátor, musíme přemístit původní rotorové vinutí DC motoru do statoru EC motoru, kde bude jednoduše napájeno. Toto vinutí je u EC motorů obvykle zapojeno do hvězdy se vzájemným posunutím o 120°. Rotor EC motoru obsahuje permanentní magnety, které jsou konstrukčně uspořádány buď na povrchu, nebo jsou vestaveny uvnitř rotoru. Permanentní magnety se vyrábějí z magneticky tvrdých materiálů, jako jsou materiály ze vzácných zemin (neodym, železo, bor), dále levnější tvrdé ferity a různé slitiny (Alnico).



Obr. 3. 1 Konstrukce EC motoru[4]

3.3 Elektronická komutace EC motoru

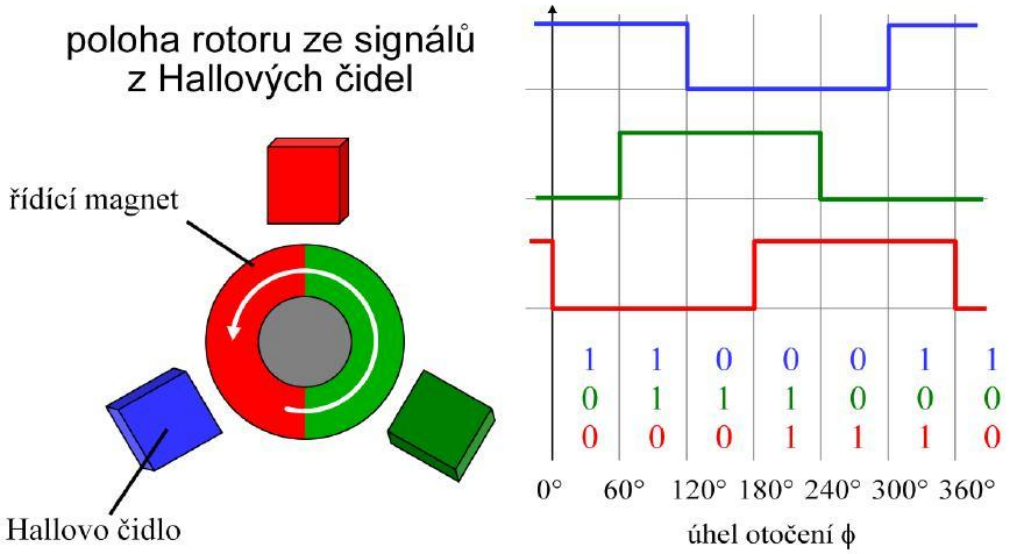
V předchozí podkapitole jsem se záměrně vyhnul důležité části EC motoru, elektronické komutaci, neboť tuto část je vhodné více rozvést. Elektronickou komutaci, tedy přepínání napájecího proudu do jednotlivých částí vinutí, zajišťují elektronické spínací obvody na základě údaje o okamžitém natočení motoru. Toto přepínání napájecího proudu je prováděno senzorově nebo bezsenzorově.

Senzorové řízení je zajištěno několika způsoby, používají se soustavy tří Hallových sond, selsyny a resolvery. Nevýhodou takového řízení je vyšší pořizovací cena. Pro komutaci statorového stejnosměrného proudu do následujícího fázového vinutí potřebujeme snímat polohu rotoru, a to diskrétně vždy po 60°. Na *Obr. 3. 2* je znázorněno rozmístění Hallových sond a tři průběhy signálů, které po dalším zpracování v logických obvodech, slouží jako povely pro komutaci tří statorových vinutí (*Obr. 3. 3*)

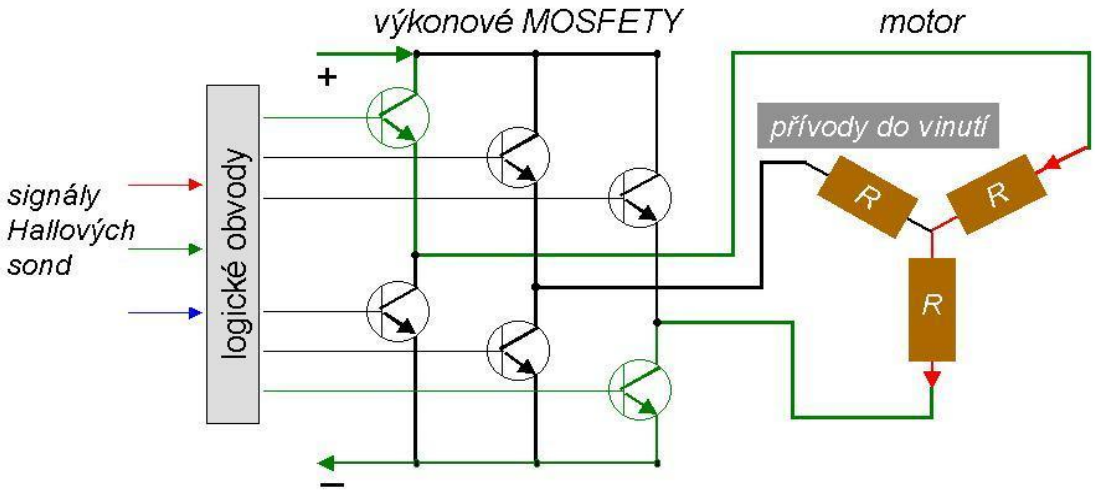
Bezsenzorové řízení má oproti senzorovému řízení výhodu v celkovém zjednodušení konstrukce motoru a ve snížení nákladů na daný aktuátor. Mezi nevýhody patří vyšší nároky na řídící algoritmus a složitější řídící elektronika. Při tomto řízení snímáme indukované napětí na vinutí motoru, kterým v daném okamžiku neprochází proud, během otáčení motoru. Musíme se ovšem vypořádat s neznalosti přesné polohy při startu motoru, proto je třeba použít komplexnější elektroniku a algoritmus pro řízení rozběhu.

Výrobců EC motorů je velká řada, ze všech motorů, které jsem na internetu našel, mě zaujal motor firmy Maxon EC6 (*obr. 3. 4*), který má v průměru jen 6 mm.

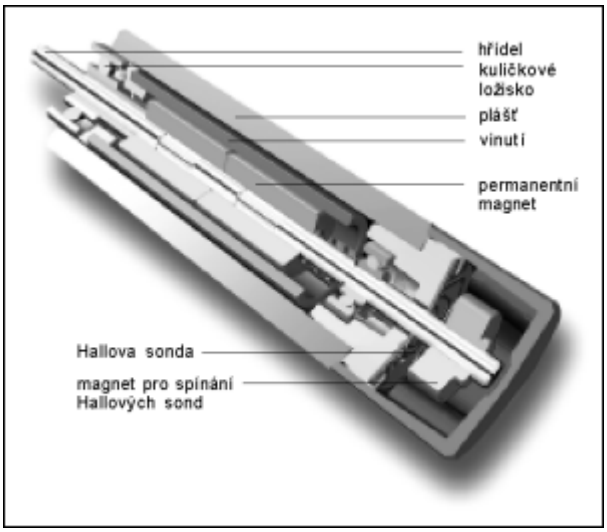
Závěrem kapitoly 3 bych chtěl podotknout, že díky absenci mechanického komutátoru mají EC motory vyšší životnost než klasické stejnosměrné motory a EC motory jsou omezeny jen životností použitých ložisek.



Obr. 3. 2 Senzorová komutace 2 pólového EC motoru[4]



Obr. 3. 3 Elektronika komutace EC motoru[5]



Obr. 3. 4 EC motor Maxon EC6[6]

4 MĚŘENÍ

Jedním z cílů této bakalářské práce bylo provést a vyhodnotit měření pomocí LabVIEW a CompactRIO na EC motoru. Základní ideou celého měření bylo, že jsme nejdříve pomocí měřícího programu získali soubor hodnot. Hodnoty jsme následně zpracovali tzv. offline, tedy bez dalšího aktuálního přísunu dat, pomocí programu pro vyhodnocení. Výsledky jednoduchých měření pomohly k ověření správnosti navrženého měřícího programu, který lze dále rozvíjet, a jehož pomocí by bylo možné provést rozsáhlejší a náročnější měření. V následujících podkapitolách je nejdříve popsán program pro měření a následně program pro zpracování.

4.1 Program pro měření

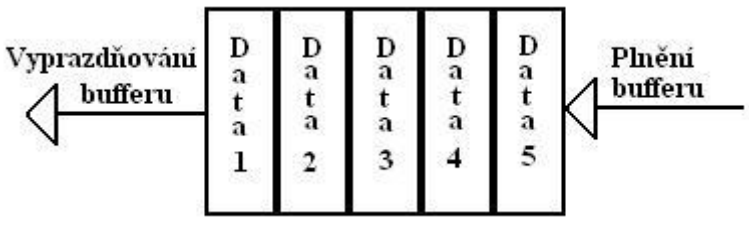
Jak vyplývá z teorie uvedené v první kapitole, měřící program ve vývojovém prostředí LabVIEW se dá rozdělit do tří částí. První částí je VI pro FPGA, následované VI řadiče Real-Time a posledním VI běžícím na počítači.

4.1.1 Synchronizace, FPGA FIFO

Už na začátku tvorby jednotlivých programů je třeba si uvědomit, že pokud chceme vytvořit funkční soustavu tří programů, mezi kterými jsou přenášena data, můžeme se potýkat s jedním vážným problémem. Tímto problémem je synchronizace. Jednotlivé programy (na hostitelském PC, na řadiči Real-Time a na FPGA) běží samy o sobě asynchronně. V programech CompactRIO jsou obvykle používány různé cykly (while loop, for loop, time loop). Jak lze vidět na obrázcích v následujících kapitolách, i program pro měření v této práci obsahuje různé cykly. Mohlo by se stát, že budou cykly pro zápis dat programu pro FPGA vykonávány rychleji, než cykly pro čtení dat programu pro řadič Real-time. Řadič Real-Time neodešle všechny data na hostitelské PC a některá data by mohla být ztracena. Proto potřebujeme ukládat data do paměti do té doby, než budou přečtena. K ukládání dat se dá použít buffer (zásobník). Buffer je oblast počítačové paměti, která skladuje vícenásobné položky dat. Abychom se vyhnuli ztrátě dat, můžeme cílový i hostitelský program synchronizovat. Synchronizace je také nutná pro řízení časování (timing control). Mezi možnosti synchronizace časováním a řízením signálů patří hodiny, spouštěče a události (clocks, triggers, events). Mezi metody synchronizace a použití bufferu můžeme zařadit:

- přerušení (interrupts);
- handshaking;
- DMA FIFO(Direct memory access).

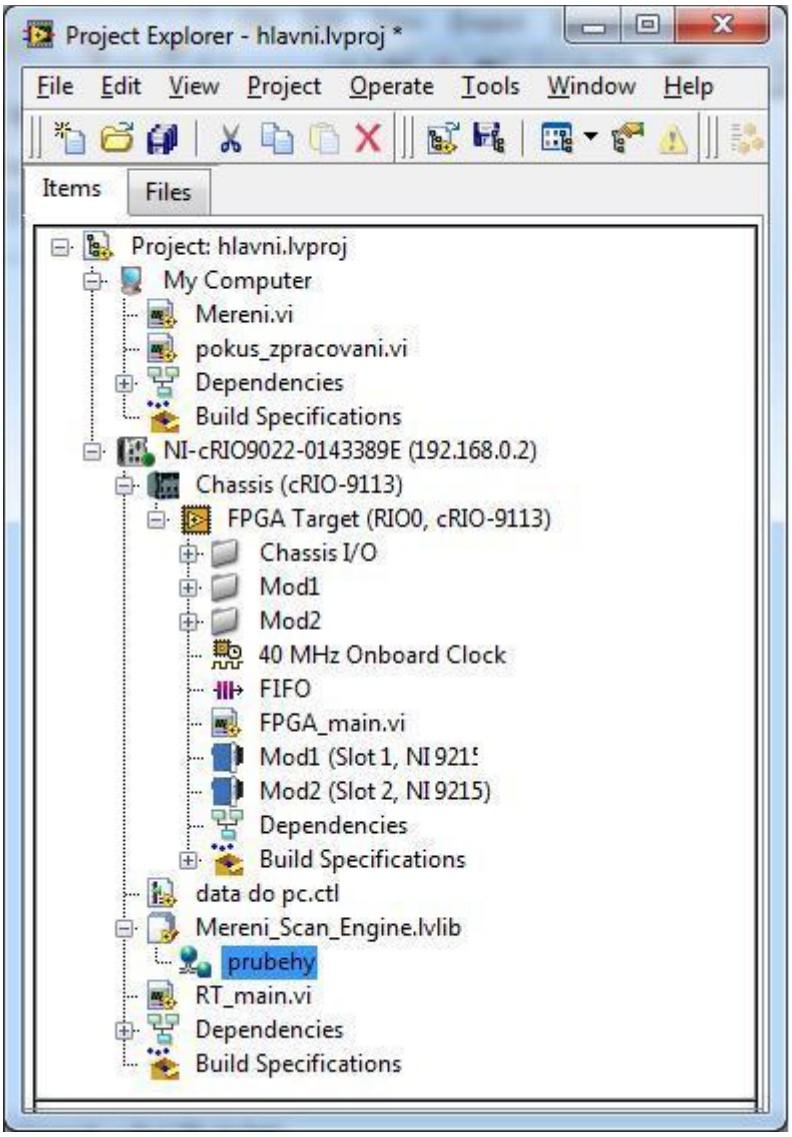
V měřícím programu této práce byla použita paměť FPGA FIFO. Paměť FIFO (First in, First out) je buffer (*Obr. 4. 1*), jehož obsahem může být jakýkoliv datový typ. Data uložená do této paměti jako první se také jako první budou z paměti číst.



Obr. 4. 1 Příklad bufferu pro 5 hodnot.

4.1.2 Tvorba projektu

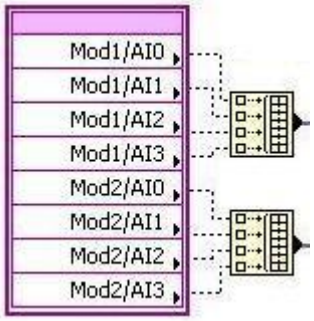
Soustava několika VI, sdílených proměnných a dalších prvků tvořících jeden celek je v LabVIEW označována jako Projekt. Nový projekt vytvoříme vybráním položky *File >> New* a *Empty Project*. Projekt jsme pojmenovali hlavni.lvproj. Do projektu musíme přidat řadič Real-Time cRIO-9022, to provedeme kliknutím pravého tlačítka myši na Project:hlavni.lvproj a vybráním položky *New >> Targets and Devices*. V dalším okně vybereme *new target or device*, rozklikneme položku *Real-Time CompactRIO* a zvolíme *cRIO- 9022*. Počkáme, až nám LabVIEW nalezne všechny I/O moduly a přidá FPGA. Dále je třeba rozkliknout *Chassis (cRIO-9113)* a *FPGA Target (RIO0, cRIO-9113)* a ověřit, jestli v projektu vidíme všechny připojené moduly (Mod1, Mod2, …). Po rozkliknutí konkrétního modulu ještě zkontrolujeme, zdali jsou v modulu všechny proměnné (AI0, AI1, …) dané počtem kanálu I/O modulu. Projekt uložíme a další části se již budou přidávat později, jak bude popsáno v následujících kapitolách. Konečná podoba projektu je na *Obr. 4. 2.*



Obr. 4. 2 Kompletní projekt

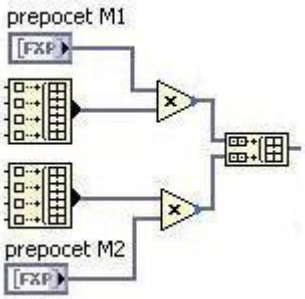
4.1.3 Program pro FPGA

Tato část měřícího programu obstarává načítání signálů z I/O modulů a dále je zpracovává a zapisuje na FIFO paměť. V projektu je potřeba vytvořit VI pro FPGA. V okně projektu klikneme pravým tlačítkem myši na *FPGA Target (RIO0, cRIO-9113)* a vybereme položku *New >> VI*, které pojmenujeme FPGA_main. V nově otevřeném VI si zobrazíme blokový diagram a začneme postupně vkládat jednotlivé bloky. Jako první potřebujeme blok pro načtení signálů z I/O modulů. Ten se jmenuje FPGA I/O Node (*Obr. 4. 3*) a nalezneme ho v paletě nástrojů *FPGA Module VIs and Functions >> FPGA I/O Functions.* V tomto programu byly použity dva I/O moduly s čtyř- kanálovými vstupy, proto je v bloku 8 proměnných Mod1/AI0 až Mod2/AI3.



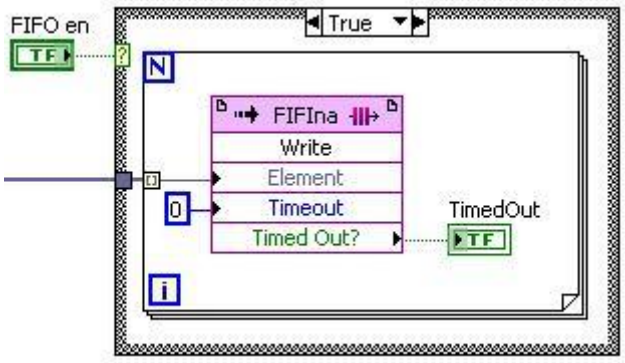
Obr. 4. 3 Block FPGA I/O Node a blok Build Array.

Aby se nám s výstupy bloku lépe pracovalo, sloučíme je po 4 kanálech do dvou matic 4x1 pomocí bloku Build Array (*Obr. 4. 3*) v paletě *Programming >> Array*. Případnou úpravu nebo přepočet hodnot každé matice nám umožní blok Multiply *(Obr. 4. 4),* v paletě P*rograming >> Numeric*. Tento blok vynásobí dvě hodnoty přivedené na jeho vstupy. Přepočítané výstupy z obou matic použitím dalšího bloku Build Array opět sloučíme do jedné matice a dostaneme tak data typu array 8x1 (*Obr. 4. 4*).



Obr. 4. 4 Přepočítaná a sloučená matice 8x1.

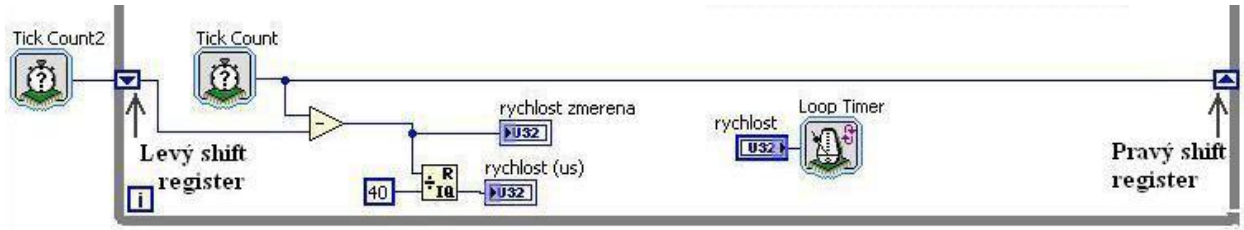
Data jsou dále přivedena do bloku Case Structure (*Obr. 4. 5*), který nalezneme v paletě *Programming >> Structures.* V bloku Case Structures vytvoříme subdiagramy. Každý subdiagram má pomocí nastavení bloku definovanou hodnotu, při níž se daný subdiagram provede. Tato hodnota je přiváděna do vstupu Case Selector, v našem případě je to proměnná *FIFO en* nabývající hodnot True a False. Do subdiagramu vložíme blok FIFO Method Node, method nastavíme na Write. Tento blok nám zapíše data z matice do FIFO paměti.



Obr. 4. 5 Blok Case Structure s blokem FIFO Write.

Abychom mohli ovládat rychlost načítaní dat z I/O modulů, musíme výše zmíněnou soustavu bloku uzavřít do cyklu while. To provedeme blokem While Loop v paletě *Programming >> Structures*. Do While Loop přidáme blok Tick Count, Tick Count2, blok Loop Timer (obojí v paletě *Programming >> Timing*), dále blok Subtract a blok Quotient & Reminder (obojí v paletě *Mathematics >> Numeric).* Ovládání rychlosti načítaní dat z I/O modulů (*Obr. 4. 6*) pak funguje nasledovně:

Bloky Tick Count a Tick Count2 začnou sčítat ticky – výstup časovače, který definuje počet period od zapnutí časovače. 1 perioda trvá 0.025µs (frekvence FPGA 40 MHz). Výstupy čítačů jsou přivedeny na vstup bloku Subtract (matematický rozdíl) a rozdíl těchto hodnot je pak zobrazen indikátorem *rychlost zmerena*. Současně je rozdíl přiveden na blok Quotient & Reminder, na jehož výstupu IQ se nám zobrazí výsledek celočíselného dělení *rychlost zmerena*/40. Tím získáme skutečnou rychlost v µs. Výstup čítače je přiveden do pravého posuvného registru (shift register). Po uplynutí doby, dané nastavením bloku Loop Timer, nám proběhne další iterace smyčky. V každé další následující smyčce se hodnota pravého shift registru přenese do levého shift registru a odečte se od hodnoty čítače Tick Count, čímž budeme vždy vědět, jakou rychlostí se nám načítají data z I/O modulů.



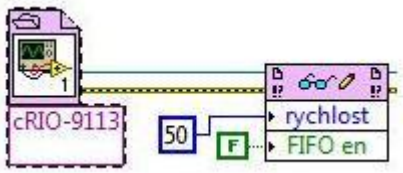
Obr. 4. 6 Ovládání rychlosti načítaní dat z I/O modulů

Kompletní program pro FPGA je v části přílohy uveden jako Příloha 1.

4.1.4 Program pro RT

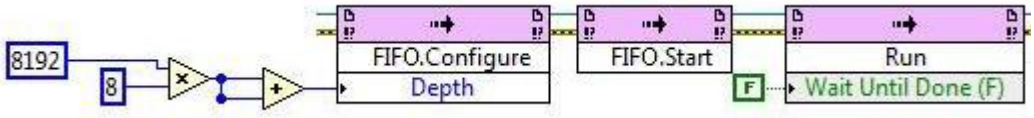
Tato část měřícího programu slouží pro komunikaci s FPGA a pro ukládání dat do sdílené proměnné. Nejdříve je potřeba vytvořit VI pro řadič Real-Time. V okně projektu klikneme pravým tlačítkem myši na NI-cRIO9022a vybereme položku *New >> VI*. Nové VI pojmenujeme RT_main. Dalším krokem bude vytvoření sdílené proměnné (Shared variables). Sdílená proměnná slouží pro komunikaci mezi PC a řadičem Real-Time. Sdílenou proměnnou vytvoříme kliknutím pravého tlačítka myši v okně projektu na My Computer vybráním položky *New >> Variable*. V nastavení proměnné, které se nám otevřelo, změníme jméno (*prubehy*) a datový typ (array of double).

Abychom mohli komunikovat mezi VI pro řadič Real-Time a VI pro FPGA, musíme použít blok Open FPGA VI Reference (*Obr. 4. 7*), který nalezneme v paletě *FPGA Interface*. Za tímto blokem bude následovat blok Read/Write control (paleta *FPGA Interface*), pomocí kterého nastavíme v FPGA_main.vi indikátor *rychlost* na 50 (frekvence vzorkování 20kHz) a proměnnou *FIFO en* na FALSE. Připomeňme si, že proměnná *FIFO en* sloužila k ovládání Case Structure v FPGA_main.vi, takže změnou hodnoty na FALSE zatím znemožníme zápis do FIFO paměti.



Obr. 4. 7 Blok Open FPGA VI Reference a blok Read/Write kontrol

Dále je potřeba nakonfigurovat FIFO paměť. Pomocí bloku Invoke Method v paletě *FPGA Interface* a vybráním FIFO.Configure můžeme nastavit velikost paměti přidáním konstanty na vstup Depth. Jako optimální hodnotu jsme stanovili 8192 vzorků * 8 kanálů * 2 (blok Multiply a blok Add) = 131072 prvků. Dalším blokem Invoke Method, vybráním FIFO.Start zapneme přenos dat. Zapojením třetího bloku Invoke Method, vybráním Run spustíme VI pro FPGA (FPGA_main.vi). Všechny 3 bloky jsou v zapojení na *Obr. 4. 8*.

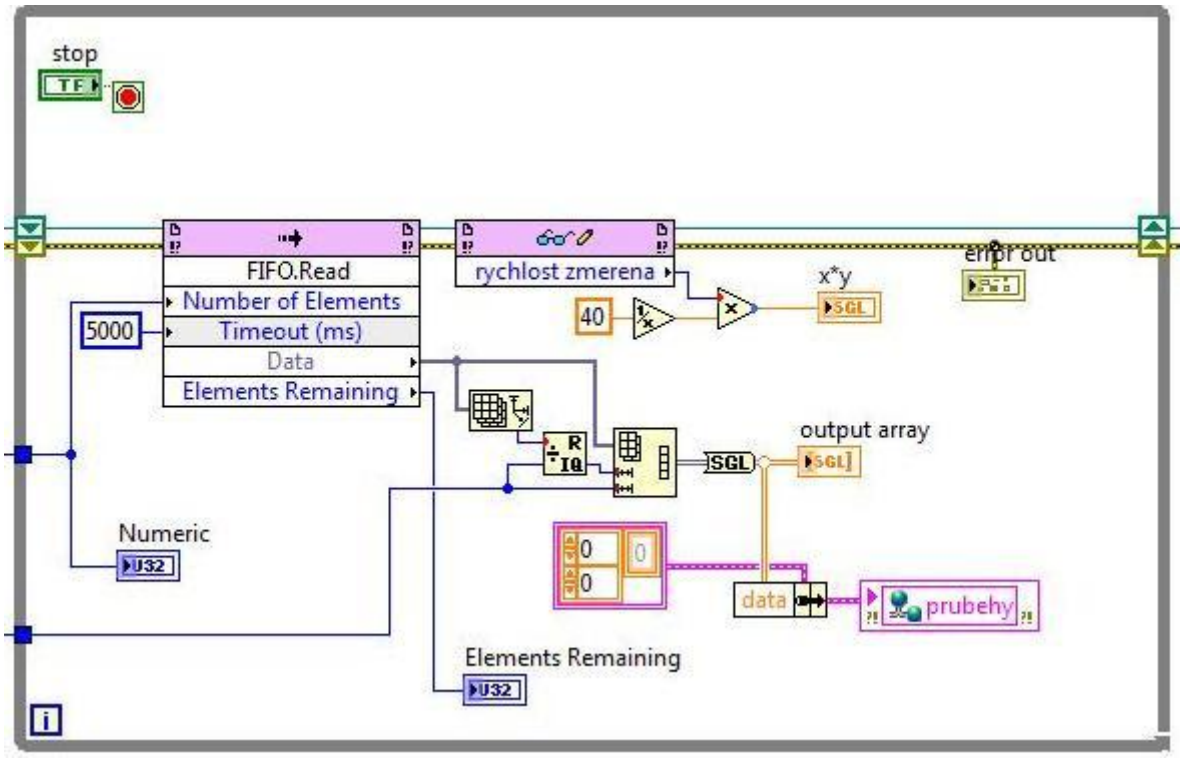


Obr. 4. 8 Bloky Invoke Method

Zapojením bloku Read/Write kontrol za bloky Invoke Method a nastavením proměnné *FIFO en* na TRUE, umožníme zápis do paměti FIFO.

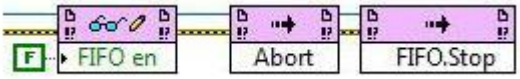
Abychom si mohli kdykoliv zastavit čtení z paměti FIFO a zápis do sdílené proměnné *prubehy,* vložíme si následující program (*Obr. 4. 9*) do cyklu while (While Loop). Protože jsme předchozími bloky spustili FPGA a povolili zápis do paměti FIFO, můžeme nyní z paměti FIFO číst (použitím bloku Invoke Method a vybráním FIFO.Read). U bloku se nastaví Number of elements (131072 prvků) a vstup Timeout (ms) ponecháme na původním nastavení 5000. Na výstupu Data se nám pak objeví hodnoty ve tvaru matice o velikosti 1xN. Na výstupu Elements remaining můžeme sledovat počet dosud nezpracovaných prvků. Pro vytvoření matice stejného

rozměru jako při čtení z I/O modulů připojíme k výstupu Data blok Reshape Array (paleta *Programming >> Array*) a dále k tomuto bloku připojíme bloky Array Size a Quotient & Reminder. Tímto zjistíme přesný počet řádků matice Nx8, který připojíme opět k bloku Reshape Array. Poté hodnoty matici Nx8 přetypujeme pomoci bloku To Single Precision Float na matici čísel s plovoucí desetinnou čárkou. Dále pro snadnější přenos přes sdílenou proměnnou vytvoříme svazek (bundle) dat blokem Bundle By Name (paleta *Programming >>Cluster, Class and Variant)*, jehož výstup je už připojen k bloku sdílené proměnné.



Obr. 4. 9 Bloky pro čtení z paměti FIFO a zápis do sdílené proměnné

Mezi posledními bloky (Obr. 4. 10) v tomto blokovém diagramu je opět blok Read/Write control k nastavení proměnné *FIFO en* na FALSE (znemožnění zápisu do FIFO) a dva bloky Invoke Method. Jeden s nastavením Abort k přerušení FPGA VI a druhý s blok s nastavením FIFO.Stop k zastavení přenosu dat a k smazání paměti FIFO.

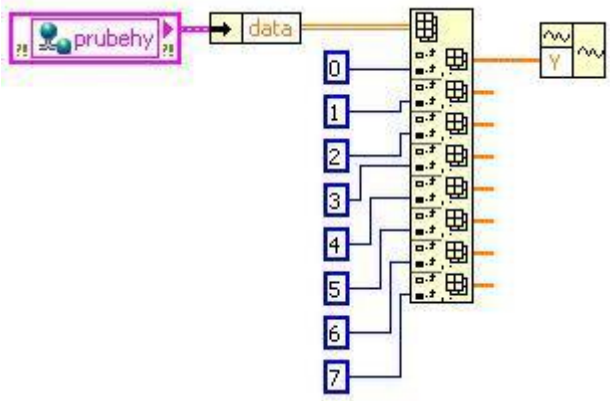


Obr. 4. 10 Bloky Invoke Method k zastavení a vymazání FIFO paměti.

Kompletní program pro Real-Time je v části přílohy uveden jako Příloha 2.

4.1.5 Program pro PC

Program na PC má za úkol načíst data ze sdílené proměnné a dále je zpracovat na konkrétní hodnoty napětí, proudu, momentu a otáček. V okně projektu klikneme pravým tlačítkem myši na My computer a vybereme položku *New >> VI*. Do nově vzniklého VI vložíme sdílenou proměnnou. Data ze sdílené proměnné *prubehy* připojíme k bloku Unbundle By Name (paleta *Programming >>Cluster, Class and Variant)*, kterým ze svazku dat, vytvoříme jednotlivá data (matici 8xN). Dále pomocí bloku Index Array rozdělíme matici na 8 matic 1xN. Z těchto matic pomocí bloku Build Waveform (Analog Waveform) vytvoříme 8 průběhů, reprezentující napětí, proud, moment a otáčky (*Obr. 4. 11*).



Obr. 4. 11 Sdílená proměnná, Bloky Unbundle By Name, Index Array a BuildWaveform.

Jen v případě průběhů napětí můžeme průběhy rovnou převést na bloky pro Fourierovu transformaci a měření RMS (obojí popsané v Kapitole 2). U ostatních průběhu musíme signál ještě přepočítat pomocí bloku Formula (*Obr. 4. 12*), který nalezneme v paletě *Express >> Arithmetic&Comparison.*

Obr. 4. 12 Blok Formula pro přepočet signálu

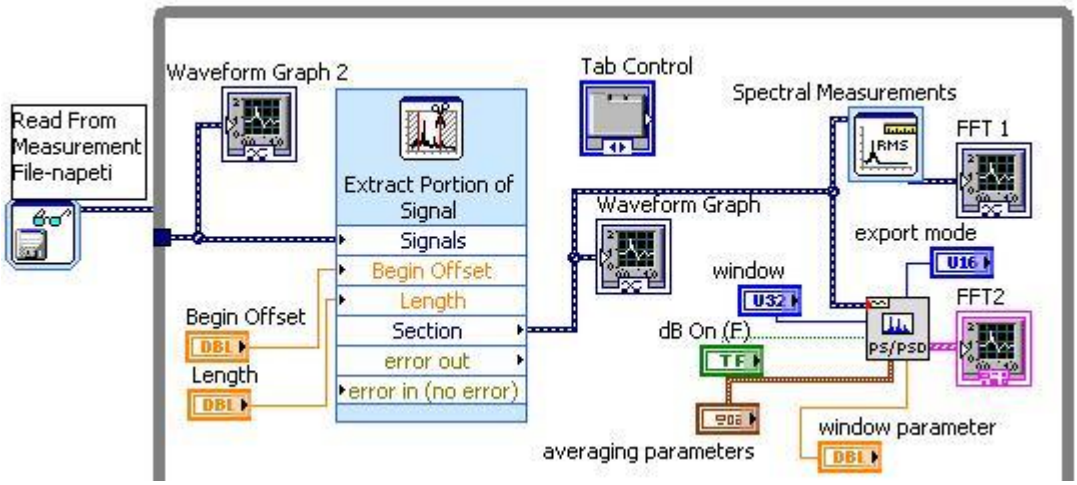
Použitím dalších bloků Formula vypočítáme průměrné napětí, průměrný proud, mechanický a elektrický výkon motoru a jeho účinnost. Program pro měření naměří a zobrazí hodnoty ve formě číselných ukazatelů a jednoduchých grafů. Abychom s hodnotami mohli dále pracovat, potřebujeme všechny hodnoty nejdříve uložit. To provedeme vložením čtyř bloků Write To Measurement File (*Obr. 4. 13*), který nalezneme v paletě *Express >> Output*. Vytvoříme tak čtyři soubory s příponou lvm, s kterými budeme pracovat v další kapitole.

Obr. 4. 13 Blok Write To Measurement File

Kompletní schéma Programu pro PC je v části přílohy uveden jako Příloha 3.

4.2 Program pro vyhodnocení

Pomocí programu pro měření jsme vytvořili čtyři lvm soubory naměřených hodnot, které si můžeme snadno načíst do dalšího VI. V okně projektu klikneme pravým tlačítkem myši na My computer a vybereme položku *New >> VI*. Pojmenovali jsme ho *Zpracovani.vi*. Do předního panelu nově vzniklého VI vložíme prvek Tab Control (*paleta Modern >> Containers*). Tím si vytvoříme přepínatelné rámečky, do kterých postupně naskládáme grafy jednotlivých průběhů. Do blokového diagramu vložíme blok Read From Feasurement File (paleta *Express >> Input*), kterým načteme hodnoty ze souboru lvm. Jelikož naměřené signály obsahují mnoho dat (myšleno mnoho period signálu) použijeme blok Extract Portion of Signal (paleta *Express >> Signal Manipulation*). Tím si zobrazíme na následně připojeném bloku pro graf jen několik period signálu. Následují bloky pro FFT a k nim zapojené bloky pro zobrazení grafů. Na *Obr. 4. 14* jsou vidět zapojené všechny výše zmíněné bloky programu pro zpracování.



Obr. 4. 14 Načítání hodnot a jejich následné zpracování

Obrázek předního panelu VI pro zpracování signálů by při nastavení okrajů tohoto dokumentu nebyl příliš čitelný, proto je přesunut do části příloh pod názvem Příloha 4.

4.3 Použité přístroje

V této kapitole jsou uvedeny použité měřicí přístroje, včetně jejich technických parametrů.

Chassis typ NI cRIO-9113

- 4-slotové nastavitelné chassis kompatibilní s jakýmkoliv CompactRIO I/O modulem
- Xilinx Virtex-5 nastavitelné I/O (RIO) FPGA jádro
- Schopnost automaticky slučovat uživatelskou kontrolu a techniku zpracování obvodu

Řadič Real-Time NI cRIO-9022

- 533 MHz procesor, 2 GB Trvalé paměti, 256 MB DDR2 paměti
- Dva Ethernetové porty s vloženými Web a FTP servery pro vzálené uživatelské rozhraní
- Vysokorychlostní USB port pro spojení USB flash pamětí a zařízení
- RS232 seriový port pro připojení periferních zařízení
- Dvojité stejnosměrné napájení 9 až 35 V

I/O modul NI9239

- 4-kanálový, 24-bitový analogový vstupní modul
- Izolace 250 Vrms mezi jednotlivými kanály
- Filtr na antialias
- ±10 V vstupní rozsah

I/O modul NI9215

- 4-kanálový, 16-bitový analogový vstupní modul
- Izolace 250 Vrms mezi jednotlivými kanály
- ±10 V vstupní rozsah
- NIST-vysledovatelná kalibrace

Všechny použité přístroje, měřící aparatura a zařízení jsou dobře vidět na fotce, která je přesunuta do Příloh pod názvem Příloha 5.

5 ZÁVĚR

Cílem práce bylo vytvořit měřící program v programovacím prostředím LabVIEW, který by byl schopen měřit parametry EC motoru. Použití programu zjednoduší měření, výsledky se dají snadno dále zpracovávat, hovoříme o digitalizaci měření. Díky programu jsme také docílili menší časové náročnosti měření.

V první kapitole bylo čtenáři představeno programové vývojové prostředí LabVIEW, technologie FPGA a její zakomponování do platformy CompactRIO.

Následující kapitola seznámila čtenáře s metodikou měření, uvedla základy měření efektivní hodnoty (RMS) a vysvětlila matematickou metodu rychlé Fourierovy transformace (FFT).

Třetí kapitola uvedla rozdělení EC motorů, popsala jejich konstrukci a vysvětlila funkci EC motoru.

Ve čtvrté kapitole naplnila práce požadavky cíle stanoveného v Úvodu. Na počátku kapitoly jsou zmíněné možné problémy při měření a návod na jejich odstranění. Postupné představování bloků potřebných pro správnou funkci programu ukázalo řešení měřícího programu. Někdy byly zmíněny i alternativní postupy použitím jiných bloků. Krátkým měřením se ukázal program „za běhu“, což potvrdilo správnou funkci programu, jak lze vidět na obrázcích uvedených v Příloze 4 a v Příloze 5.

LITERATURA

[1] ŠŤASTNÝ, Jakub. *FPGA prakticky: Realizace číslicových systémů pro programovatelná hradlová pole*. Praha: Ben-technická literatura, 2010. 200 s.

[2] *CompactRIO™ and LabVIEW™ Development Fundamentals Course Manual: Course Software Version 8.6 September 2008 Edition Part Number 324516C-01* [online]. Austin, Texas: National Instruments, 2008 [cit. 2011-05-20]. Dostupné z WWW: <http://kantorp.sweb.cz/cRIO_8-6_eng.pdf>.

[3] JURA, Pavel. *Signály a systémy: Část 3: Diskrétní signály a diskrétní systémy* [online]. Brno: FEKT VUT Brno, 2010 [cit. 2011-05-22]. Dostupné z WWW: <https://www.vutbr.cz/www_base/priloha.php?dpid=33242>.

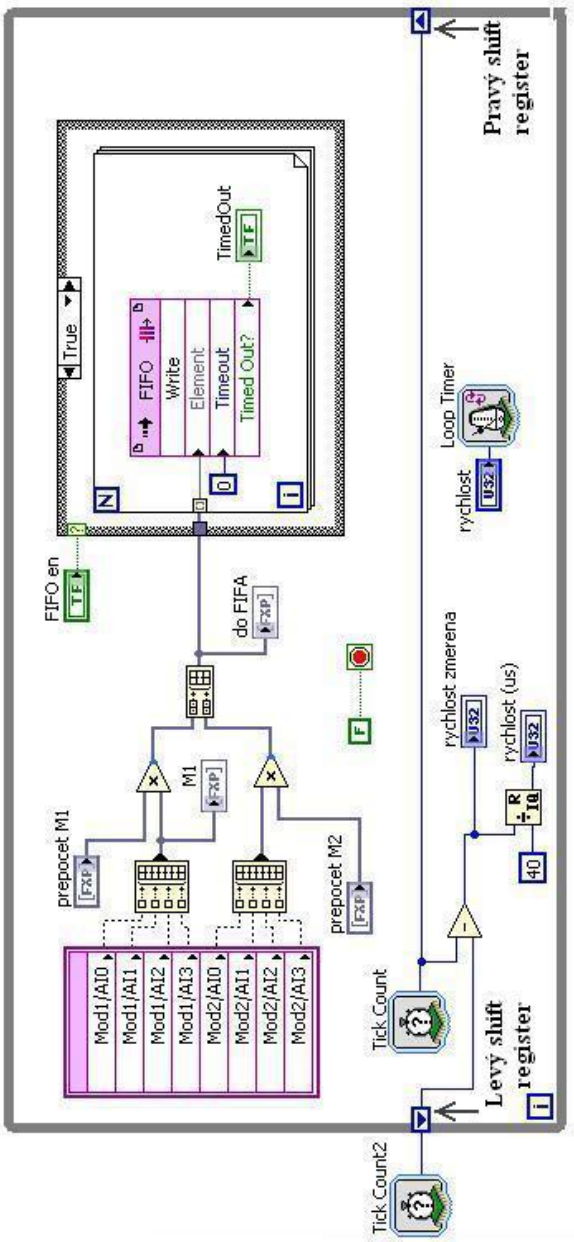
[4] HUZLÍK, Rostislav. *Jaja.kn.vutbr.cz* [online]. 2008 [cit. 2011-05-26]. EC motory. Dostupné z WWW: <http://jaja.kn.vutbr.cz/~huzlik/EC%20motor.pdf>.

[5] *Uzimex.cz* [online]. 2002 [cit. 2011-05-24]. Malé stejnosměrné motory Maxon. Dostupné z WWW: <http://www.uzimex.cz/soubory/20070103_maxon_serial.pdf>.

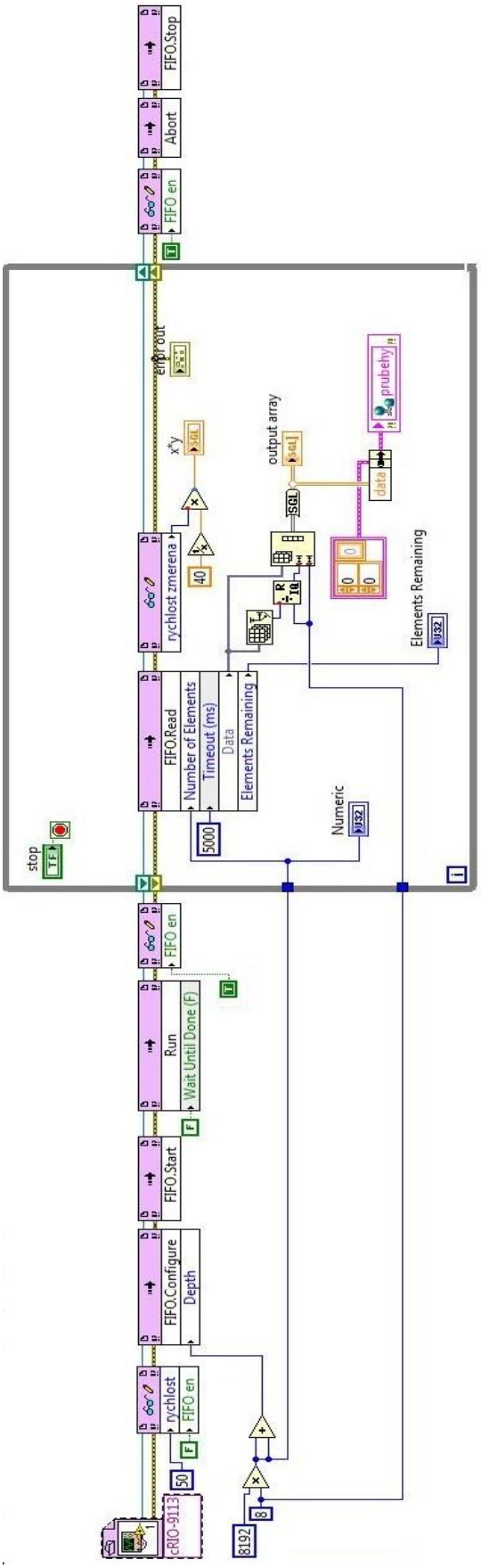
[6] BROŽ, Václav. *Automa* [online]. 2010 [cit. 2011-05-30]. Nová řada malých stejnosměrných motorů EC. Dostupné z WWW: <http://www.odbornecasopisy.cz/index.php?id_document=27736>.

SEZNAM PŘÍLOH

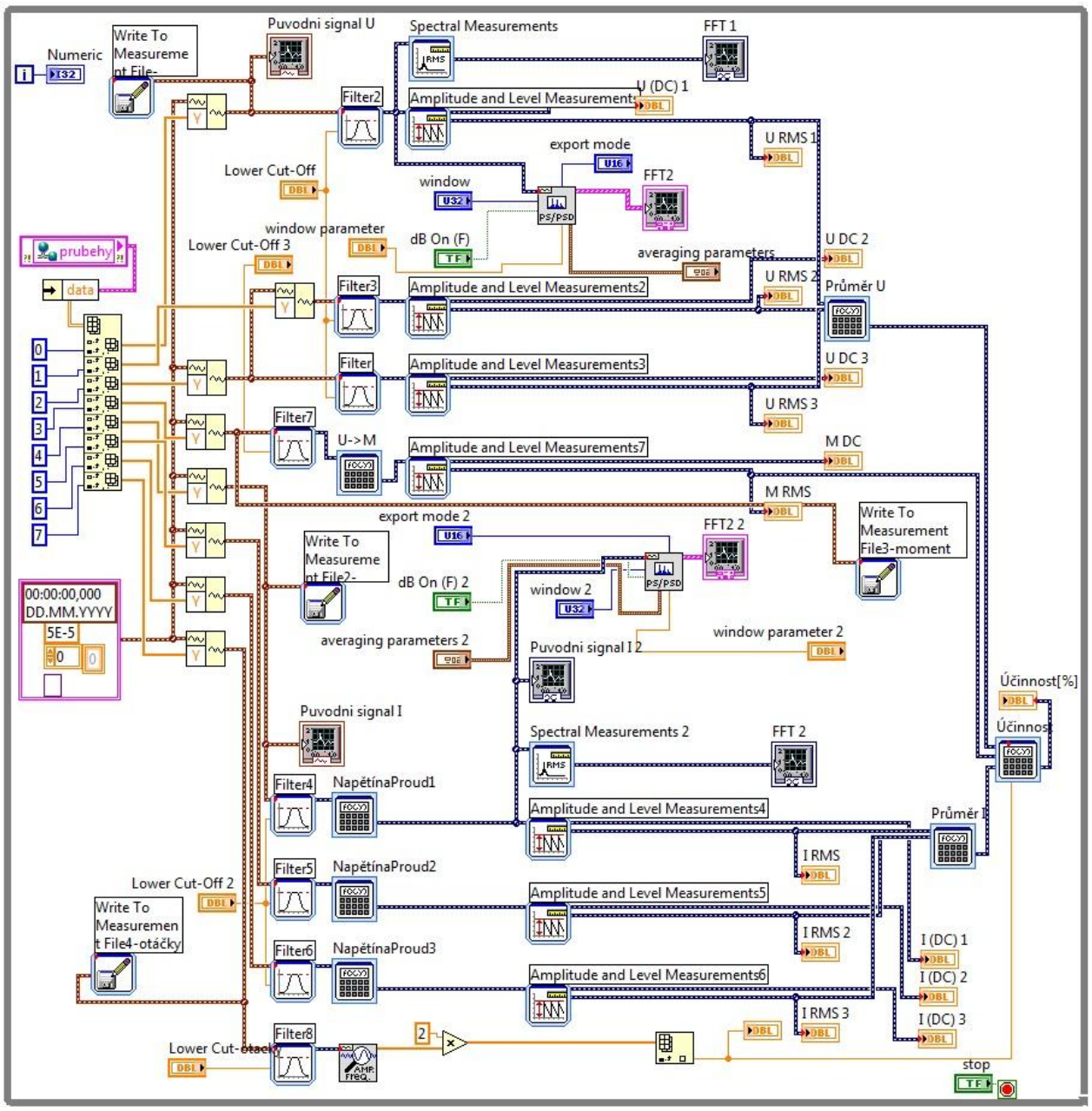
Příloha 1: program pro FPGA



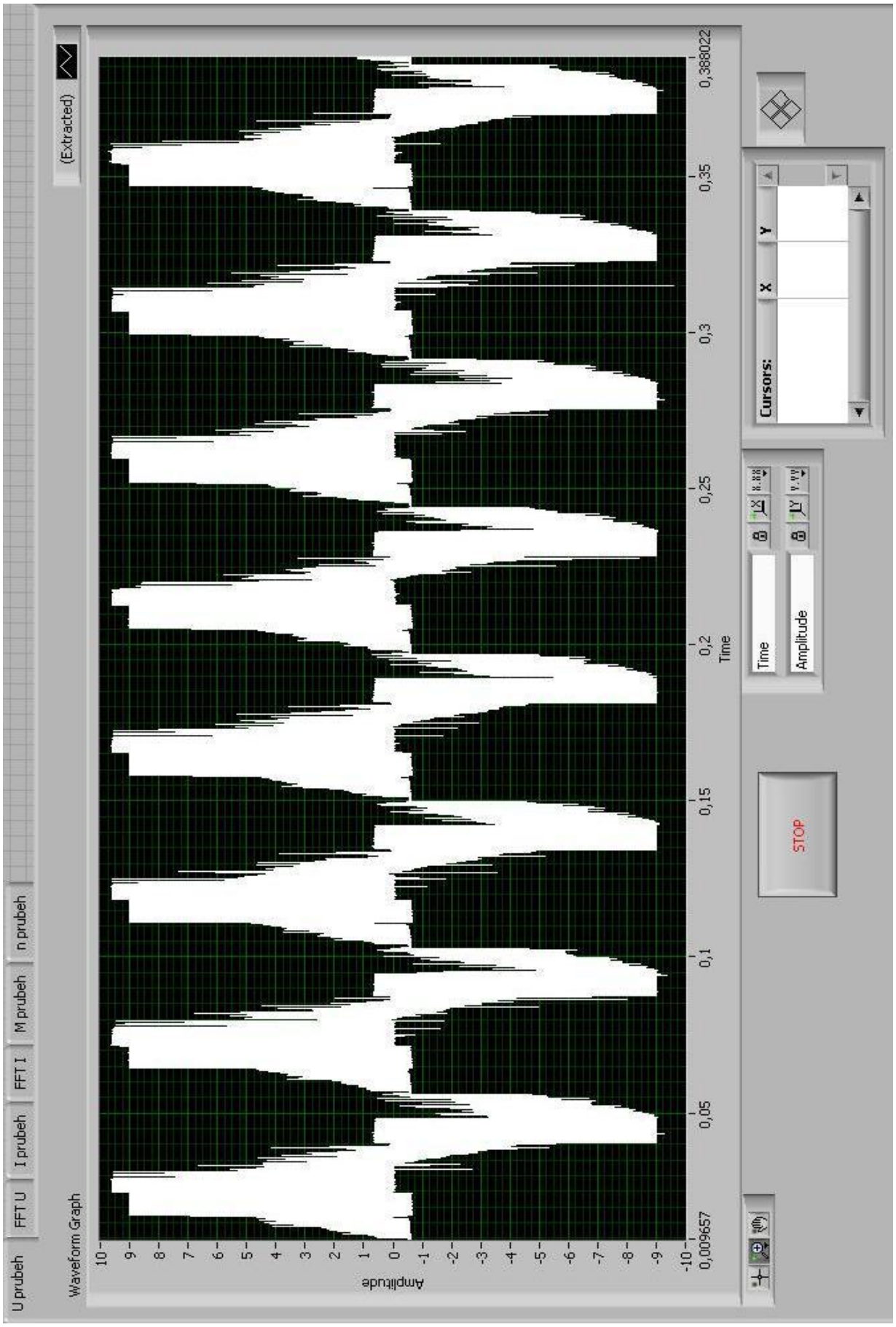
Příloha 2: Program pro Real-Time



Příloha 3: Program pro PC



Příloha 4: Program pro vyhodnocení



Příloha 5: Měřící stanoviště