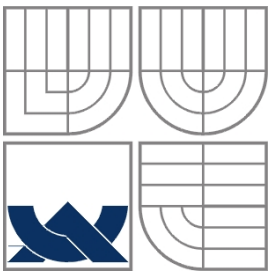


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

# SOFTWARE PRO ŘÍZENÍ EXPERIMENTŮ NA BIOMECHANICKÉM TESTOVACÍM ZAŘÍZENÍ.

THE SOFTWARE FOR EXPERIMENTS CONTROL ON BIOMECHANICAL MACHINE.

**DIPLOMOVÁ PRÁCE**
MASTER`S THESIS

**AUTOR PRÁCE** **Bc. TOMÁŠ HEJČ**
AUTHOR

**VEDOUCÍ PRÁCE** **Ing. VÍT ONDROUŠEK**
SUPERVISOR

BRNO 2009

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2008/2009

# ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Tomáš Hejč

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Aplikovaná informatika a řízení (3902T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

**Software pro řízení experimentu na biomechanickém testovacím zařízení.**

v anglickém jazyce:

**The Software for Experiments Control on Biomechanical Machine.**

Stručná charakteristika problematiky úkolu:

Práce je zaměřena na realizaci nejvyšší vrstvy software. Tento software musí umožnit plánovat, spouštět a zaznamenávat experimenty prováděné na zařízení sloužící pro testování vlivu zatížení na velikost otěru páteřních prvku.

Cíle diplomové práce:

1) Stručně popište testovací zařízení a objasněte hlavní princip fungování stroje.
2) Shrňte požadavky kladené ze strany uživatelů na plánování experimentu, sběr dat a jejich zobrazování.
2a Navrhněte a popište hlavní algoritmus nejvyšší vrstvy tohoto software.
2b Navrhněte a popište stěžejní nástroje pro realizaci pomocí NI LabVIEW (paralelní běh, synchronizace, přenos dat mezi vlákny, ukládání velkého objemu dat)
3) Návrh realizujte. Zaměřte se na popis závažných problému, které bylo nutné při realizaci řešit.
4) Ověřte funkčnost realizovaného software při dlouhodobých testech (nejméně pět testů trvajících 8 hodin). V případě potřeby navrhněte patřičné modifikace software. Vyhodnoťte navržené řešení.

Seznam odborné literatury:

www.ni.com

Vedoucí diplomové práce: Ing. Vít Ondroušek

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2008/2009.

V Brně, dne 10.10.2008

L.S.

| | |
|---|---|
| doc. RNDr. Ing. Miloš Šeda, Ph.D. | doc. RNDr. Miroslav Doupovec, CSc. |
| Ředitel ústavu | Děkan fakulty |

## ABSTRAKT

Cílem této diplomové práce je návrh a realizace uživatelského rozhraní v prostředí NI LabVIEW pro konstrukčně zdokonalený prototyp biomechanického testovacího přístroje realizovaného v rámci spolupráce ÚMTMB a ÚAI. Jedná se o přístroj zkoumající vlastnosti páteřních obratlů. Hlavním úkolem práce je realizovat software, který umožní plánovat, spouštět a zaznamenávat dlouhodobé experimenty na tomto zařízení.

Práce je členěna na část teoretickou a praktickou. První z nich se zabývá popisem testovacího přístroje, dále jsou shrnuty veškeré kladené požadavky ze strany uživatelů, následuje popis použitých nástrojů pro realizaci software v prostředí NI LabVIEW 8.6. Praktická část se zabývá samotným návrhem a realizací vytvořeného software pro plánování a řízení experimentů vyvinutého v prostředí NI LabVIEW 8.6 a jeho následným testováním. Závěrem je rozebrán vznik a řešení stěžejních problémů vzniklých při realizaci.



## ABSTRACT

The main aim of this master's thesis is the design and the realization of graphical user interface for constructionally-advanced prototype of biochemical device, which was realized in terms of the cooperation of ISMMB (Institute of Solid Mechanics, Mechatronics and Biomechanics) and IAI (Institute of Automation and Computer Science). The NI LabVIEW developing suite has been used as the main design tool of this software. This experimental device is used to investigate characteristics of spinal vertebra. The main aim of this thesis is to realize the control software, which is able to plan, initialize and record the long-term experiments on this device.

This thesis is divided into theoretical and practical part. First part is focused on the description of the experimental device and sums all apointed requirements. This part also includes characterization of software tools, that has been used to realization in NI LabVIEW 8.6 development suite. The practical part deals with the design and successive realization of the developed application used for planning and controlling experiments, and also describes performed long-term tests. The solutions of a few fundamental problems, which were shown during these tests, are also included.



## KLÍČOVÁ SLOVA

Dlouhodobé testování, grafické programování, NI-PCI6220, NI-SCC-68.

## KEYWORDS

Long-term testing, graphical programming, NI-PCI6220, NI-SCC-68.

## PODĚKOVÁNÍ

Nemalé poděkování za pomoc a správné vedení při tvorbě této práce patří vedoucímu panu Ing. Vítu Ondrouškovi. Také bych rád poděkoval Ing. Pavlu Houškovi, Ph.D. za pomoc a poskytnutí informací potřebných k práci se zařízením zkoumající vliv otěru na páteřních prvcích. Všem blízkým a přátelům, kteří mi svoji trpělivostí a tolerancí vytvořili podmínky pro tvorbu této závěrečné práce.

**Obsah:**

**ZADÁNÍ DIPLOMOVÉ PRÁCE** ..... **3**
**ABSTRAKT** ..... **5**
**KLÍČOVÁ SLOVA** ..... **5**
**PODĚKOVÁNÍ** ..... **7**

**Část A – Teoretická část**

**1 ÚVOD** ..... **11**
**2 TESTOVACÍ ZAŘÍZENÍ** ..... **13**
2.1 Popis testovacího zařízení ..... 13
2.2 Elektromechanická část ..... 14
2.3 Použité prostředky na zjištění momentů ..... 15
**3 POUŽITÝ SOFTWARE** ..... **17**
3.1 Použité vývojové prostředky ..... 17
3.2 Nejvyšší vrstva řídicího software ..... 17
3.2.1 Software pro nejvyšší vrstvu ..... 17
**4 POŽADAVKY NA SOFTWARE KLADENÉ ZE STRANY UŽIVATELE** ..... **19**
4.1 Požadavky na řídicí software ..... 19
4.2 Požadavky na software pro offline reprezentaci dat ..... 20
4.3 Požadavky na software pro převod dat ..... 20
**5 POPIS STĚŽEJNÍCH NÁSTROJŮ PRO REALIZACI** ..... **23**
5.1 Měření a práce s přístroji ..... 23
5.2 Nástroje pro synchronizaci ..... 26
5.2.1 Funkce Rendezvous ..... 27
5.2.2 Funkce Occurrences ..... 28
5.3 Práce s daty pomocí TDMS ..... 29
5.4 Zásobník FIFO – fronta ..... 29
5.5 Lokální proměnná ..... 31
5.6 Globální proměnná ..... 31
5.7 Nástroje pro použití .NET kódu ..... 32
5.7.1 Funkce .NET Containers ..... 32

**Část B - Praktická část**

**6 REALIZACE NÁVRHU ŘÍDICÍHO SOFTWARE** ..... **33**
6.1 Uživatelské rozhraní ..... 33
6.1.1 Uživatelské rozhraní řídicího software ..... 33
6.1.2 Uživatelské rozhraní software pro offline zobrazování dat ..... 35
6.1.3 Uživatelské rozhraní software pro konverzi dat ..... 36
6.2 Blokové schéma ..... 37
6.2.1 Blokové schéma řídicího software ..... 38
6.2.1.1 Inicializace ..... 40
6.2.1.2 Vzniklé události ..... 41
6.2.1.3 Realizace chodu pohonů ..... 45
6.2.1.4 Ukončování ..... 47
6.2.1.5 Shrnutí realizace paralelního běhu řídicího software ..... 48

6.2.2 Blokové schéma software pro offline zobrazování dat ..... 48
6.2.3 Blokové schéma software pro konverzi dat..... 49
**7 KALIBRACE ..... 51**
7.1 Použitý přístup..... 51
**8 ŘEŠENÍ PROBLÉMŮ VZNIKLÝCH PŘI REALIZACI ..... 55**
8.1 Hierarchie ukládání ..... 55
8.2 Synchronizace smyček pohybu motorů a ukládání dat ..... 55
8.3 Ukládání velkého objemu dat..... 55
8.4 Problémy vzniklé při dlouhodobých testech vytvořeného software..... 56
8.4.1 Návrh a oprava software na základě výsledků dlouhodobých testů..... 56
**9 ZÁVĚR ..... 57**

**SEZNAM POUŽITÝCH OBRÁZKŮ..... 59**
**SEZNAM POUŽITÉ LITERATURY ..... 61**
**SEZNAM PŘÍLOH ..... 63**

**Část A – Teoretická část**

# 1 ÚVOD

Konec dvacátého století je charakteristický rychlým vývojem počítačové technologie. Takový vývoj hardware a software vytvořil možnosti pro hledání odpovědí na velmi rozsáhlou problematiku týkající se biomechanických problémů, která je řešena pracovištěm ÚMTMB na fakultě strojního inženýrství VUT v Brně více než patnáct let. Pozornost byla soustředěna převážně na klinickou praxi. Byla navázána spolupráce s několika nemocnicemi v Brně (nemocnice U sv. Anny, traumatologická nemocnice, pediatrická nemocnice, fakultní nemocnice Brno - Bohunice), s několika nemocnicemi v Praze a s několika zdravotními středisky v ostatních městech nacházejících se v České republice. Vedle zdravotních zařízení byla navázána spolupráce i s několika výrobci zdravotnických pomůcek. Díky těmto spolupracím a pokroku ve výpočetní technice byl vytvořen základ pro tvorbu zařízení určených pro potřeby zdravotní péče. [1]

V rámci spolupráce ÚMTMB a ÚAI v návaznosti na klinickou praxi jsou vyvinuty dva přístroje určené k dlouhodobému testování otěru páteřních prvků. Důvodem tvorby druhého zařízení bylo zjištění konstrukčních nedostatků. Oba přístroje jsou osazeny třemi stejnosměrnými motory umožňující řídit pohyb vzorku ve třech osách. Studie je založena na snaze simulovat fyziologický pohyb páteřních elementů tj. simulovat mechanické namáhání vzorku v oblasti torze (krut) a flexe (ohyb). [2]

Software umožňující řízení zmíněných pohonů a tudíž vytvářející pohyb elementů lze rozčlenit do několika vrstev. Nejnižší je spjata s výkonovými obvody umístěnými na zařízení. Nadřazená vrstva, implementující řídicí algoritmy, je vystavěna na *.NET Framework*. V rámci předešlé práce tvořené jako bakalářská práce byly vytvořeny v programovém prostředí NI LabVIEW 8.2 tři samostatné software reprezentující nejvyšší vrstvu. Tyto vytvořené software jsou určeny pro konstrukčně neodladěný prototyp, viz. [3]. Vytvořené software tvořily základní pilíř znalostí o dané problematice pro tvorbu této diplomové práce. Tato diplomová práce je zaměřena na tvorbu software reprezentujících nejvyšší vrstvu v programovém prostředí NI LabVIEW 8.6 s ohledem na požadavky pracovníků ÚMTMB. Programové prostředí NI LabVIEW bylo vybráno mimo jiné z důvodu snahy o standardizaci a umožnění spolupráce s průmyslem a výměny výsledků mezi dalšími pracovišti, která na tomto projektu spolupracovala, jako Florida International University USA. Návrh software je určen pro konstrukčně zdokonalený prototyp přístroje určeného ke zkoumání velikosti otěru na páteřních prvcích. Navazuje a značně rozšiřuje předešlou bakalářskou práci. Cílem nejvyšší vrstvy je umožnit zaznamenávání a plánování experimentů a interpretaci naměřených dat. Vytvořený software proto musí zahrnovat uživatelské rozhraní umožňující všechna potřebná nastavení. Diplomová práce je rozdělena na teoretickou a praktickou část. Jednotlivé části jsou děleny pomocí logicky řazených kapitol.

Teoretická část je pojata jako seznámení s problematikou a použitými prostředky pro realizaci. Je zde uveden popis konstrukčně zdokonaleného prototypu zařízení sloužícího pro zkoumání velikosti otěru na páteřních prvcích, dále popis prostředků zajišťujících snímání působících momentů na ose flexe. V teoretické části jsou také obsaženy použité vývojové prostředky, kde je popis zaměřen hlavně na nejvyšší vrstvu, a seznámení s použitým programovým prostředím NI LabVIEW. Jsou zde shrnuty požadavky ze strany pracovníků ÚMTMB, na jejichž základě bylo po vzájemných konzultacích vytvořeno optimální prostředí software určené pro uživatele k plánování

dlouhodobých experimentů. Teoretická část je ukončena popisem stěžejních nástrojů použitých při realizaci software.

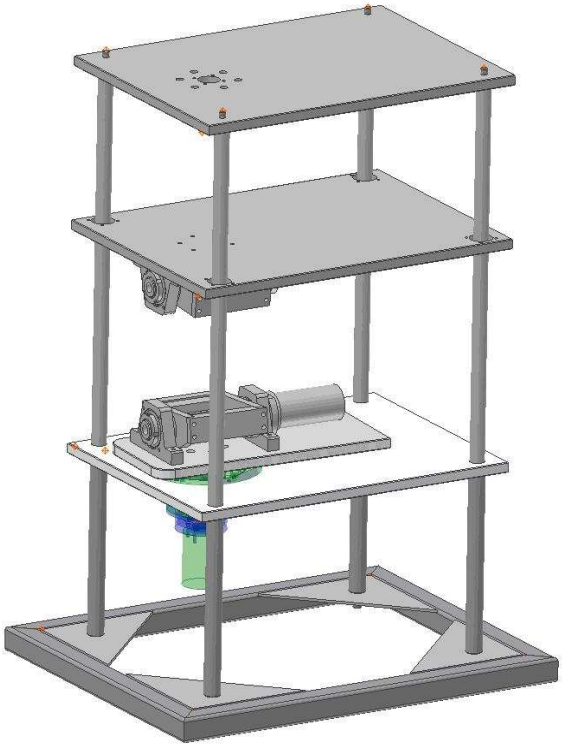
Praktická část této diplomové práce představuje vlastní práci. Zabývá se konkrétním řešením vytvořeného software v prostředí NI LabVIEW 8.6. Jsou zde rozebrány jednotlivé kroky vývoje a hlavní myšlenky návrhu tohoto software a popsány funkce, které nabízí vytvořený software uživateli při provádění dlouhodobých experimentů. Praktická část obsahuje také popis přístupu k řešení problémů s kalibrací senzoru pro snímání napětí umožňujícího zjištění velikosti momentu v ose flexe. Jsou zde vyčteny problémy vzniklé při realizaci software. Závěrem jsou popsány výsledky a závěry dlouhodobých testů, které byly provedeny na zařízení z důvodů ověření funkčnosti při dlouhodobém provozu.

# 2 TESTOVACÍ ZAŘÍZENÍ

## 2.1 Popis testovacího zařízení

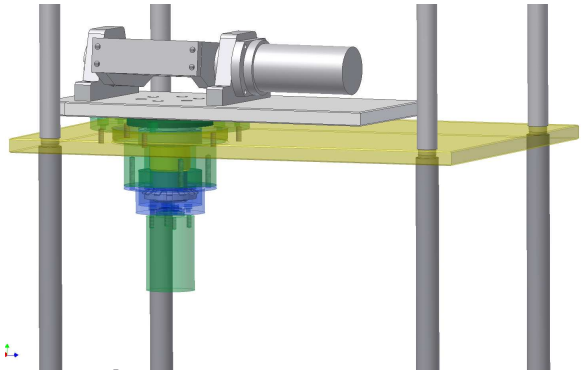
Jak bylo zmíněno v předcházející kapitole, v současné době jsou vytvořena dvě zařízení určená pro dlouhodobé testy na páteřních vzorcích. Přepracované a zhotovené zařízení je možno vidět na obr. 1. Na základě zkušeností z předešlého řešení byla provedena náhrada stávajících pohonů za pohony, které vyhovují řešené úloze. Na základě výsledků průzkumu trhu byly pro jednotlivé osy zvoleny pohony od firmy Maxon. Došlo ke konstrukční úpravě, viz. obr. 3 a 4, uchycení pohonů a nahrazení původně použitých třecích spojek v osách flexe spojkami přenášejících moment přes pero. Do zařízení byly zakomponovány tenzometry. Poslední konstrukční úpravou bylo přesunutí snímače síly pro měření zatížení vzorku ze svislé středové osy přípravku do svislé osy procházející středem upínacích misek přípravku obr. 2. [2]

*Obr. 1 Konstrukčně zdokonalené zařízení pro zkoušení páteřních segmentů.*

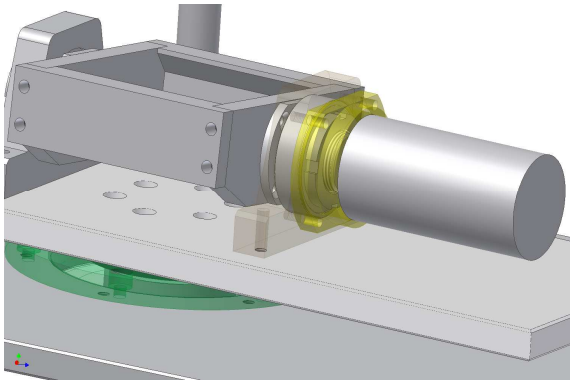


*Obr. 2 Celkový pohled na inovovanou sestavu zařízení pro zkoušení páteřních segmentů. [2]*

Páteřní vzorek testovaný na otěr je nutné upravit z důvodu správného upnutí, proto dojde k zalití dvou protilehlých ploch vzorku Durakrylem. Takto upravený vzorek se vloží do misek testovacího zařízení, viz. obr. 4. [4]



*obr. 3 Detail uchycení motoru pohonu* torze. *[2]*



*Obr. 4 Detail uchycení motoru pro* flexi. *[2]*

## 2.2 Elektromechanická část

Elektromechanická část zařízení obsahuje tři pohybové osy. Osy flexe i torze jsou osazeny výrobky od firmy Maxon, viz. tab. 1 a tab. 2. Všechny tři osy jsou tvořeny stejnosměrným komutátorovým motorem typu RE 36, čtyřstupňovou planetovou převodovou skříní GP42C a magnetickými inkrementálními snímači MR typu L snímačem otáček s rozlišením 1024 dílků na otáčku, viz. obr. 1. Motory jsou připojeny přes výkonové obvody do COM portu PC. Na hřídeli spojující motory flexe a misek jsou upevněny tenzometry, pomocí nichž je přepočítáván moment, viz. kap. 7.1.

Tab. 1: Osy flexe [2]

| **Nominální hodnoty motoru RE 36** | |
|---|---|
| napájecí napětí | 24 V |
| kroutící moment | 0,0782 Nm |
| otáčky | 92.2 $s^{-1}$ |
| **Hodnoty převodové skříně GP42C** | |
| maximální trvalý výstupní moment | 15 Nm |
| převodový poměr | 936:1 |
| **Snímač otáček MR ENC typ L** | |
| dílků na otáčku | 1024 |
| počet kanálů | 3 |
| **Parametry pro řízení** | |
| otáčkové předimenzování | 3.8x |
| minimální řiditelné otáčky | 0.001 $s^{-1}$ |
| $T_{IRC}$ | 2 ms |
| maximální provozní otáčky motoru | 24.5 $s^{-1}$ |
| minimální řiditelné provozní otáčky motoru | 0.977 $s^{-1}$ |
| maximální počet pulsů IRC za $T_{IRC}$ | 51 |

Tab. 2: Osa torze [2]

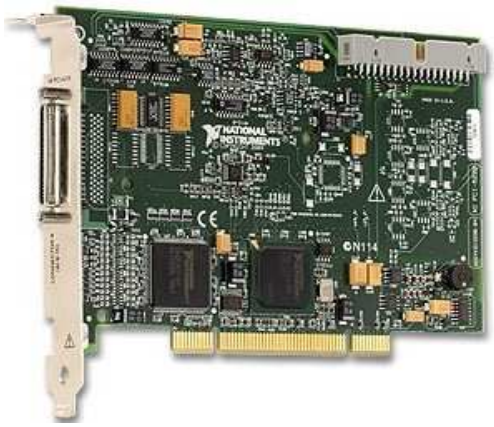
| **Nominální hodnoty motoru RE 36** | |
|---|---|
| napájecí napětí | 24 V |
| kroutící moment | 0,0782 Nm |

| | |
|---|---|
| otáčky | 92.2 $s^{-1}$ |
| **Hodnoty převodové skříně GP42C** | |
| maximální trvalý výstupní moment | 15 Nm |
| převodový poměr | 546:1 |
| **Snímač otáček MR ENC typ L** | |
| dílků na otáčku | 1024 |
| počet kanálů | 3 |
| **Parametry pro řízení** | |
| otáčkové předimenzování | 3.8x |
| minimální řiditelné otáčky | 0.00179 $s^{-1}$ |
| $T_{IRC}$ | 2 ms |
| maximální provozní otáčky motoru | 23.8 $s^{-1}$ |
| minimální řiditelné provozní otáčky motoru | 0.977 $s^{-1}$ |
| maximální počet pulsů IRC za $T_{IRC}$ | 51 |

## 2.3 Použité prostředky na zjištění momentů

Pomocí tenzometrů umístěných na hřídeli mezi motory a miskami na ose flexe se přenáší signál do zařízení - terminálu od firmy National Instruments (dále NI) SCC-68, viz. obr. 5. Tenzometry jsou připojeny pomocí SCC slotu a pro přesnější měření je přiváděno na SCC-68 z externího zdroje napětí o velikosti 5 V. Prostřednictvím karty NI PCI-6220 zobrazené na obr. 6 je zpracováván signál z SCC-68. Karta PCI-6220 nabízí 24 digitálních I/O a 32-bit převaděč. Terminál a karta jsou spojeny stíněným 68-pin kabelem SHC68-68-EPM.

*Obr. 5 Terminál NI SCC-68. [5]*



*Obr. 6 Karta NI PCI-6220. [5]*

# 3 POUŽITÝ SOFTWARE

## 3.1 Použité vývojové prostředky

Tato práce vychází z navrženého a dále pak částečně realizovaného řešení, v němž je pro vývoj software použito několika vývojových prostředků, jiných pro každou vrstvu řídicího software. Nejnižší vrstva je spjata s výkonovými obvody umístěnými na zařízení a je tvořena ve vývojovém prostředí MS Visual Studio 2005 v jazyce C++. Prostřední vrstva implementující řídicí algoritmy je tvořena opět ve vývojovém prostředí MS Visual Studio 2005, ale je naprogramována v jazyku C# a využívá technologii MS *.NET Framework*, což je platforma nezávislá na operačním systému. Pro běh této platformy je nutné nainstalovat *.NET Framework*. Nejvyšší vrstva je tvořena ve vývojovém prostředí NI LabVIEW 8.6. Pro běh aplikací na PC bez nainstalovaného NI LabVIEW je nutná instalace NI LabVIEW Runtime Engine. [3]

## 3.2 Nejvyšší vrstva řídicího software

Pro prostřední vrstvu vytvořenou v jazyce C# je stěžejní především implementace řídicích algoritmů pohonů. Využívá nejnižší vrstvu a poskytuje interface pro vrstvu jí nadřazenou. Právě pro tyto účely byly Ing. Pavlem Houškou, Ph.D. vytvořeny knihovny *.dll. Software pro řízení experimentu se odkazuje a bezprostředně využívá právě tyto knihovny. Pomocí nástrojů na podporu MS *.NET Framework* obsažených v NI LabVIEW je možno s těmito knihovnami pracovat. Program NI LabVIEW byl v bakalářské práci zvolen z důvodu velmi dobré podpory standardů a mezinárodní spolupráce. [3]

V rámci bakalářské práce byla rozpracována v programovém prostředí NI LabVIEW 8.2 nejvyšší vrstva. Tvorba diplomové práce navazuje na předešlou bakalářskou práci, viz. kap. 1. Došlo k razantnímu rozšíření a zvětšení robustnosti kódu, viz. kap. 4. Z důvodu návaznosti obou prací byl zvolen stejný programovací nástroj NI LabVIEW.

Jak se ukázalo v bakalářské práci [3] integrace kódu napsaného v jazyce C# a NI LabVIEW do jediné aplikace je možná a přináší celou řadu výhod, např. možnost podpory *.NET Containers*, díky níž lze použít pro interface programu *windows forms*.

Tato práce je zaměřena na vývoj nejvyšší vrstvy řídicí aplikace, proto jí je věnována samostatná kapitola, viz. kap. 3.

### 3.2.1 Software pro nejvyšší vrstvu

Jak bylo již předesláno v předchozí kapitole, pro tvorbu nejvyšší vrstvy byl zvolen programovací nástroj LabVIEW 8.6 od společnosti National Instruments. Myšlenka tvorby, na níž stojí efektivita LabVIEW uvedeného na trh v roce 1986 pro platformu počítačů Macintosh, vznikla ve skupině tvůrců pod vedením Jeffa Kodovskeho. Tato myšlenka vychází právě z úvahy, že kdo obvykle ví, co měřit, jak analyzovat a prezentovat data, je technik, který však nemusí být sám zkušeným programátorem. Své představy předává programátorovi ve formě blokového schématu. Programátor toto schéma převádí do zvolené syntaxe programovacího jazyka, což je činnost poměrně zdlouhavá a nepřináší do procesu žádné nové informace. Cílem vývojového prostředí LabVIEW je to, aby blokové schéma bylo koncovým tvarem aplikace, které se již dále nebude převádět do textové podoby. [6]

LabVIEW (Laboratory Virtual Instruments Engineering Workbench) je obecným vývojovým prostředím s bohatými knihovnami pro vytváření aplikací zaměřených do oblasti měření ve všech fázích tohoto procesu tj. sběru, analýzy a prezentace naměřených dat. Podporuje všechny čtyři základní způsoby sběru dat do počítače (z měřících přístrojů přes rozhraní RS 232 nebo GPIB, ze zásuvných multifunkčních karet a ze systému na bázi VXI sběrnice). Poskytuje uživateli plnohodnotný programovací jazyk se všemi odpovídajícími datovými a programovými strukturami v grafické podobě tzv. G jazyk (Graphical language). LabVIEW je vývojovým prostředím na úrovni např. C jazyka, ale na rozdíl od něj není orientován textově, ale graficky. [7]

Pro vytvoření nejvyšší vrstvy byla použita nejnovější dostupná verze LabVIEW 8.6 umožňující návrh a vývoj řídicích, testovacích a vestavěných systémů pomocí grafického vývojového prostředí. Vychází z přirozeného paralelního charakteru grafického programování, které uživatelům umožňuje využívat přínosů vícejádrových procesorů, programovatelných hradlových polí (FPGA) a bezdrátové komunikace. LabVIEW 8.6 obsahuje více než 1 200 pokročilých analytických funkcí optimalizovaných pro rychlejší matematické výpočty a zpracování signálu u vícejádrových systémů pro řídicí a testovací aplikace. Umožňuje lépe identifikovat paralelní sekce kódu pomocí nové funkce, která reorganizuje diagramy LabVIEW. [5]

# 4 POŽADAVKY NA SOFTWARE KLADENÉ ZE STRANY UŽIVATELE

Při tvorbě této diplomové práce se vycházelo přednostně z požadavků kladených ze strany uživatele. Tyto požadavky vznikaly ze zkušeností a vzájemných konzultací s pracovníky ÚMTMB (dále jako uživatel), při tvorbě software byly průběžně doplňovány a začleňovány.

Software musí umožnit plánovat, spouštět a zaznamenávat experimenty. Bezporuchová funkce přístroje má být zajištěna podle požadavků uživatele dvě hodiny. Toto omezení vyplynulo z faktu, že testovaný páteřní prvek postupem času mění své mechanické vlastnosti a to z důvodu osychání. Při testování software však byl tento čas předimenzován na osm hodin a to z důvodu zaručení uživatelské jistoty při probíhajících testech.

## 4.1 Požadavky na řídicí software

Pro zajištění určitého stupně uživatelského komfortu bylo provedeno několik konzultací s uživatelem. Na jejich základě byla vytvořena široká škála uživatelských možností. Software musí umožnit plánovat a spouštět experimenty (požadavek již z bakalářské práce). Uživatel musí mít možnost běžící experiment pozastavit a tím mít přístup ke změnám nastavení experimentu:

- Konečný počet cyklů, které má zařízení vykonat,
- Jméno uživatele provádějící experiment,
- Editace popisu experimentu,
- Změna rychlosti ukládání, zobrazováni (grafy) dat,
- Nastavení krajních pohybů pohonů a jejich rychlostí.

Druhou možností změny nastavení je přímo za chodu přístroje. Software musí vykazovat určitou úroveň zabezpečení tzn. ochranu proti nastavením, která mohou vyvolat kolizi (problematika řešena v bakalářské práci). Případné problémy za běhu experimentu musí být uživateli dány na zřetel a podle jejich možných následků je nutná ochrana testovacího zařízení. Při spuštění experimentu se musí uložit veškeré nastavení a informace o experimentu. K uložení těchto informací dochází i v případě, když uživatel experiment pozastaví nebo mění parametry testu při probíhajícím experimentu. Za informace o nastavení bylo uznáno po konzultacích s uživatelem vhodné uložit:

- Nastavení obsažené v *.Net Containers* pro každý z pohonů. U každého z pohonů se jedná o 70 položek nastavení,
- Popis experimentu vytvořený uživatelem,
- Jméno uživatele provádějícího experiment,
- Konečný počet cyklů, které má zařízení vykonat,
- Časový záznam, kdy experiment začal, skončil, popřípadě byl pozastaven (provedena změna nastavení),
- Maximální natočení a rychlost chodu jednotlivých pohonů, které si uživatel zadal.

Při běhu přístroje se musí uživatelem v předem stanovených časových okamžicích ukládat pro jednotlivé pohony požadovaná data:

- Úhel natočení misky se vzorkem,
- Silový moment působící na hřídeli pohonu vyvozený vzájemným třením vzorků,
- Číslo aktuálního pracovního cyklu,

- Přesný časový okamžik, v němž došlo k sejmutí dat.

Zároveň musí probíhat online vykreslování závislosti měřených dat:

- Momentů na pohonech (1), (2) a (3), úhly natočení pohonů (1), (2) a (3) v závislosti na čase,
- Momentů na pohonech (1), (2) a (3), úhly natočení pohonů (1), (2) a (3),
- Momentů na pohonech (1, 2, 3) v závislosti na čase,
- Úhly natočení pohonů (1, 2, 3) v závislosti na čase.

Velikost periody ukládání dat je nastavitelná dle požadavku uživatele. Nejnižší možná perioda je 250 milisekund a nejvyšší 2500 milisekund. Z důvodu jedinečnosti každého testu a náročné přípravy každého z experimentů je nutné zabezpečit ukládaná data. Způsob ukládání musí být robustní a odolný proti náhodným neočekávaným chybám nejen experimentu, ale i operačního systému a uživatele zařízení. Uživatel musí mít možnost volby, jak zjistit působící moment při testování páteřního prvku. Základní avšak méně přesnou možností je výpočet, který je realizován v již vytvořených knihovnách .dll. Přesnějším způsobem zjištění momentů je zjištění pomocí senzoru. Jedním z dalších požadavků je možnost testování páteřního prvku pouze v oblasti flexe tzn. umožnit, aby uživatel byl schopen vyřadit činnost pohonu namáhající páteřní prvek na torzi. Posledním ze zmíněných požadavků je umožnění nastavení referenčních poloh pohonů na počátku testování a možnost nulování momentů.

## 4.2 Požadavky na software pro offline reprezentaci dat

Při návrhu modulu pro reprezentaci dat bylo v rámci této práce vycházeno z předchozí verze modulu navrženého v rámci bakalářské práce. Základní verze umožňovala zobrazení uložených dat, přičemž byly ukládány v každý časový okamžik pouze polohy misek a velikosti působících momentů. Uložená data (momenty, polohy) bylo možné zobrazit jako výčet v tabulce, nebo pomocí grafů různých kombinací změřených veličin.

Z důvodu nových požadavků na množství ukládaných dat bylo nutné tento již vytvořený software zcela přepracovat a umožnit uživateli spouštět software pro offline reprezentaci dat z řídicího software. Jelikož při experimentu je uživateli umožněno pomocí vytvořených funkcí software měnit informace o testu a při každé takové změně dojde k jejich uložení, je nutné, aby software nově umožňoval i zobrazení změněných údajů:

- Popis experimentu vytvořený uživatelem,
- Jméno uživatele provádějícího experiment,
- Konečný počet cyklů, které zařízení mělo vykonat,
- Čas/datum/rok ve kterém byl experiment zahájen, pozměněn a kdy byl ukončen.

I když uložený soubor obsahuje i jiná data, jak bylo uvedeno v kap. 4.1, cílem tohoto software dle požadavků uživatele je výčet pouze výše zmíněných údajů a jejich reprezentace jak formou grafu, tak i tabulky.

## 4.3 Požadavky na software pro převod dat

Z důvodu zakomponování nových výše uvedených požadavků muselo dojít k rozsáhlému přepracování řešení vytvořeného v rámci bakalářské práce. Po vzájemné konzultaci s pracovníky ÚMTMB bylo shledáno, že převedená data do univerzálního formátu *.txt budou rozdělena. Data se automaticky rozdělí do dvou souborů, které obsahují nastavení experimentu a data měřená ze zařízení. Soubor obsahující data má 224 položek o nastavení

experimentu. Tento soubor je vytvořen pro potřebu případného opakování testu a musí tudíž obsahovat:

- Obecné informace o testu (jméno uživatele, popis experimentu, počet zátěžných cyklů, začátek/konec experimentu),
- Parametry pohonů při testu (úhel natočení misky se vzorkem, rychlost),
- Pro každý z pohonů informace o nastavení:
  - Pohonu,
  - Převodovky,
  - IRC senzoru,
  - Napěťového, teplotní a momentového senzoru,
  - Kalibrace senzoru,
  - Proudového, rychlostního a polohového kontroléru.

Takto vytvořené dva soubory *.txt, bylo nutné dle požadavku uživatele náležitě popsat a seřadit obsažená reprezentovaná data, viz. kap. 6.1.3.

# 5 POPIS STĚŽEJNÍCH NÁSTROJŮ PRO REALIZACI

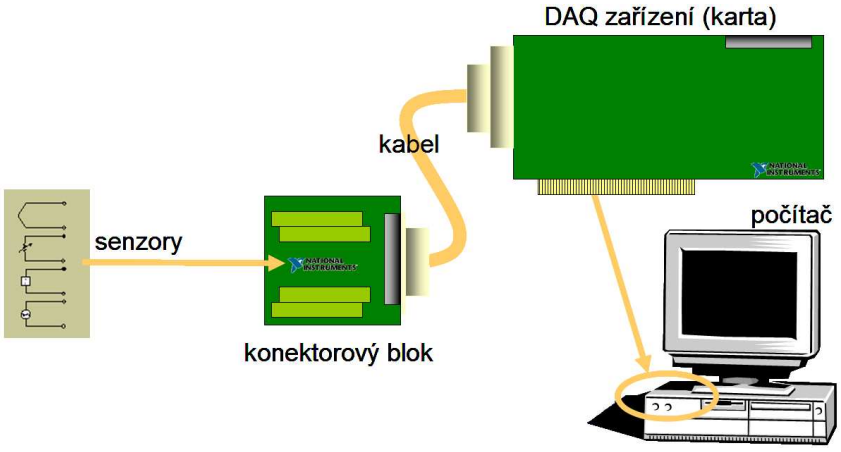
Při tvorbě této diplomové práce byla využita řada pokročilých nástrojů, které nabízí prostředí NI LabVIEW. Tato práce byla tvořena ve verzi NI LabVIEW 8.6. Tímto není zaručeno, že všechny tyto použité nástroje jsou obsaženy u předchozích verzí. Za pomoci těchto nástrojů se řešila celá řada stěžejních problémů, které vznikaly při realizaci software. Mezi takové nástroje patří:

- Expresní funkce použité pro měření a komunikaci s měřící kartou,
- Nástroj pro synchronizaci paralelního běhu smyček programu,
- Ukládání a práce s daty ve formátu *.tdms,
- Zásobník FIFO,
- Nástroje pro použití .NET kódu (vč. grafických formulářů *windows forms*) pod NI LabVIEW,
- Lokální/globální proměnné.

## 5.1 Měření a práce s přístroji

S více než 50 miliony I/O kanály prodanými za posledních 10 let je National Instruments vedoucí společností na celosvětovém trhu v počítačovém sběru dat (*DAQ*) s celou kompletní rodinou produktů *DAQ* a vylepšenou verzí *DAQmx* pro stolní, přenosné, průmyslové a embedded aplikace. Umožňuje výběr ze široké palety sběrnic a parametrů zahrnující USB, PCI, PCI Expres, PXI, PXI Express, bezdrátové sítě Ethernet. Disponuje ovladači pro různé operační systémy včetně operačního systému Windows, Linux, Mac OS X a Real-Time. Příklad zapojení využívající *DAQ* zařízení je znázorněn na obr. 7. [5]

Tradiční *NI-DAQ* je používáno v případě, že zařízení není podporováno *NI-DAQmx* softwarem (např. ATE série multifunkčních karet). Pokud je použita starší verze programu National Instruments, viz. tab. 3, nebo pokud jsou k dispozici starší programy využívající *NI-DAQ* 6.9x. *NI-DAQmx* je rozhraní vhodné pro programování analogových, digitálních I/O a ovládání stovek multifunkčních *DAQ* zařízení. NI LabVIEW obsahuje Mesurement & Automation Explorer, *DAQ* Assistant a Logger Lite software. [8]




*Obr. 7 Měřící sestava využívající DAQ kartu. [8]*

Výhody nového *DAQmx* oproti tradičnímu *DAQ* jsou [5]:

- Vylepšený stavový model,

- Multivláknové řízení,
- Odolnost v mimořádných podmínkách,
- Zjednodušená synchronizace,
- Zvýšení přehlednosti kódu,
- Umožnění přechodu k dokonalejším funkcím při zachování původního návrhu.

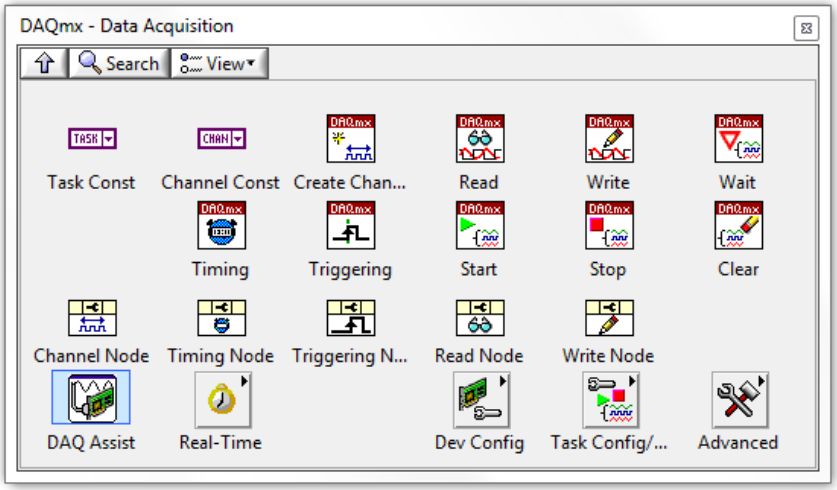
Tab. 3: Podpora pro *DAQmx* [5]

| **NI použitý software** | **Verze podporované NI-DAQmx** |
|---|---|
| LabVIEW | 7.1, 8.0, 8.2, 8.5 a 8.6 |
| LabVIEW Real-Time Module | 7.1 a novější |
| LabWindows/CVI | 7.x a novější |
| Measurement Studio | 7.x a novější |
| LabVIEW SignalExpress | 2.x a novější |

Pro práci a využití možností, které nabízí *DAQmx* od vývojářské firmy National Instruments, vytvořila tato společnost jedenáct funkcí [5]:

- DAQ Assistant,
- NI-DAQmx Create Virtual Channel,
- NI-DAQmx Trigger,
- NI-DAQmx Timing,
- NI-DAQmx Start Task (použita v diplomové práci),
- NI-DAQmx Read (použita v diplomové práci),
- NI-DAQmx Write,
- NI-DAQmx Wait Until Done,
- NI-DAQmx Stop Task (použita v diplomové práci),
- NI-DAQmx Properties,
- NI-DAQmx Clear Task.

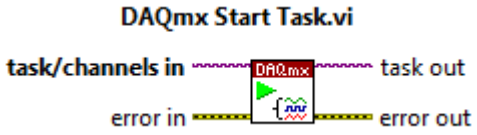
Každá z těchto funkcí je vhodná pro určitý typ aplikace. *NI-DAQmx* šetří čas a zlepšuje výkonnost aplikací tím, že poskytuje Application Programming Interface - API. Všech jedenáct funkci nám nabízí NI LabVIEW 8.6 >> Measurement I/O/DAQmx**,** viz. obr. 8. [5]




*Obr. 8 Paleta funkcí pro práci s DAQmx.*

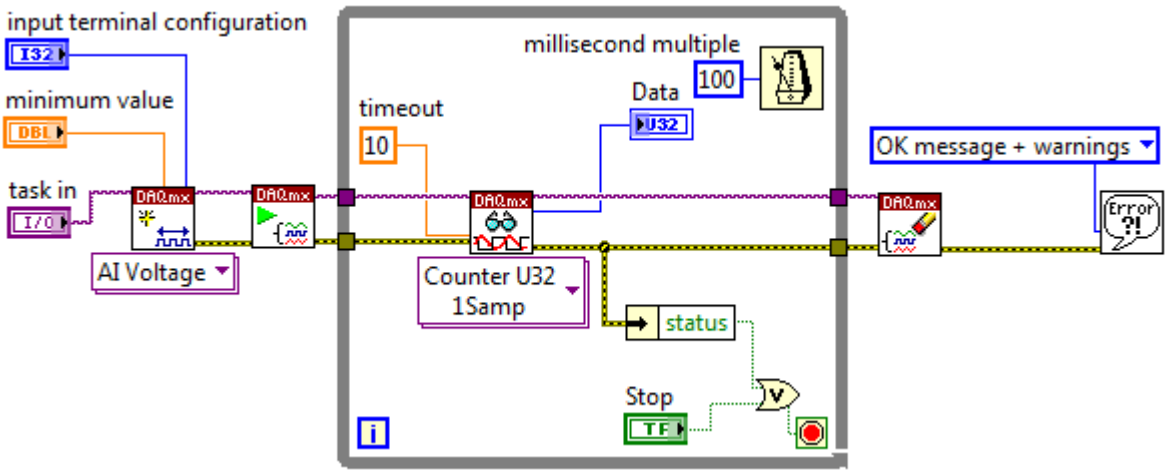
## NI-DAQmx Start Task

Předtím, než je čten nebo zapsán úkol, je nutno na začátku *DAQmx* nastavit do běžícího stavu. Jsou případy, kdy není nutné použití této funkce, viz. kap. 5.1.2. Obecně je přesto vhodné *DAQmx Start Task* používat. *DAQmx Start Task* (Measurement I/O>>DAQmx Data Acquisition palette) se nastavuje do běžícího stavu a tím dochází k měření nebo generování. Pokud není použita funkce, viz. obr. 9 měření začne automaticky, když se *DAQmx Read* spustí.




*Obr. 9 Funkce DAQmx Start Task.*

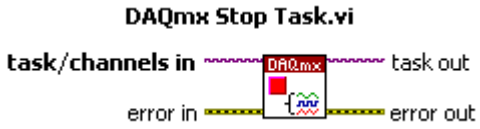
Pokud by nebyly použity funkce *DAQmx Start Task* a *DAQmx Stop Task* v případě využití *DAQmx Read,* popřípadě *DAQmx Write* a to opakovaně tzn. v smyčce, tak se začátek úkolů opakovaně zastavuje. Opětovné odstartování a ukončování úkolu zhoršuje výkon aplikace. Z tohoto důvodu je vhodné použití funkce *DAQmx Start Task,* jak je také znázorněno na obr. 10. [9]




*Obr. 10 Použití funkce DAQmx Start Task. [5]*

## NI-DAQmx Stop Task

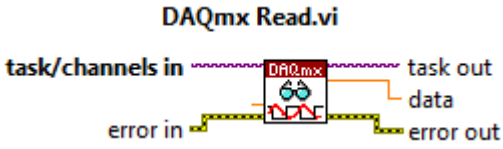
Funkce *NI-DAQmx Stop Task,* viz. obr. 11 se používá ve spojení s *DAQmx Start Task.* Pokud je explicitně voláno *DAQmx Start Task,* začne se provádět tvorba kanálu a přepne se ze stavu „nečinný“ do „běžící“. *NI-DAQmx Stop Task* slouží pro opětovný návrat běžícího kanálu do stavu „nečinnosti“. Pokud by došlo k pokusu volat *DAQmx Start Task* dvakrát za sebou, bez přerušení prvního úkolu bude vygenerována chyba. V případech konečného sběru dat v procesech *DAQmx Read, Stop Task* mohou nastat případy, kdy není zapotřebí používat tyto funkce. Problém je řešen za pomocí *NI-DAQmx* ovladačů. [5]




*Obr. 11 Funkce DAQmx Stop Task.*

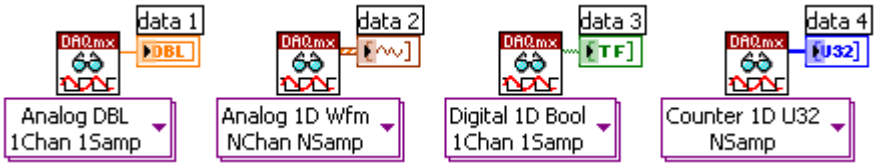
### NI-DAQmx Read

*NI-DAQmx Read* čte vzorky pro zvolený typ dat. Různé typy možných vstupních funkcí umožňují volbu dat (analogové, digitální), množství virtuálních kanálů, počet vzorků a datový typ reprezentovaných dat. Na obr. 12 je znázorněna funkce *NI-DAQmx Read*. [5]




*Obr. 12 Funkce DAQmx Read.*

Na obr. 13 jsou zobrazeny pro názornost čtyři příklady použití různých instancí *NI-DAQmx Read*. Jedná se o funkce schopné číst více vzorků připojených na vstupu. Funkce *NI-DAQmx Read*, která má na obr. 13 výstupní „data 1“, čte vzorek s pohyblivou desetinnou čárkou z úkolu, který obsahuje jen jeden analogový vstupní kanál. Funkce s výstupními „data 2“ čte časový průběh signálu po jednotlivých vzorcích z každého kanálu daného úkolu, který obsahuje jeden nebo více analogových vstupních kanálů. Pro funkci „data 3“ platí, že čte pole booleovských hodnot z úkolu, který obsahuje jeden digitální vstupní kanál. Poslední funkci zobrazenou na obr. 13, která má na výstupu „data 4“, čte jedno nebo více 32-bit celých čísel bez znamének a používá instance, pomocí kterých *NI-DAQmx* vrací upravený vzorek stejně jako pro událost sčítání. Funkce čekají na všechny požadované vzorky, které mají být získávány, a pak začnou tyto vzorky číst. [10]




*Obr. 13 Čtyři možnosti použití DAQmx Read.*

## 5.2 Nástroje pro synchronizaci

Pokud jsou v programovém prostředí NI LabVIEW vytvořeny dva datově na sobě nezávislé cykly, není běh těchto cyklů vzájemně synchronizován, takže se časem začne rozcházet počet oběhů prvního a druhého cyklu vlivem toho, že náhodně oba tyto cykly sdílejí kapacitu procesoru s jinými úlohami, které souběžně běží v rámci operačního systému. V případě nutnosti synchronizace je nutné využít některý z nástrojů, které nabízí prostředí NI LabVIEW. [7]

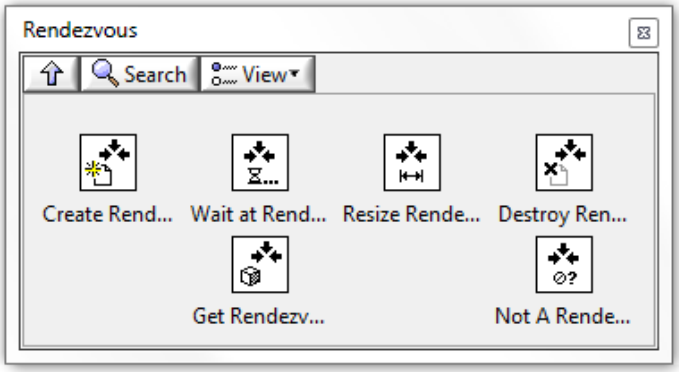
V této diplomové práci bylo využito dvou nástrojů pro synchronizaci běhu cyklů. Prvním zmíněným a zároveň nejvíce stěžejním nástrojem v rámci diplomové práce je

*Rendezvous*. Zabezpečuje chod hlavních smyček programu, viz. kap. 6.2.1.5. Druhým použitým v rámci synchronizace je nástroj *Occurrences*, který se nachází ve smyčce, která řídí běh pohonů, viz. kap. 6.2.1.3.

### 5.2.1 Funkce Rendezvous

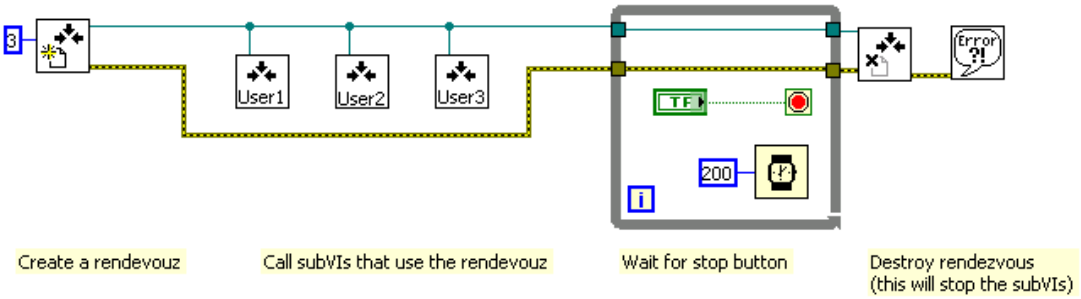
Pomocí této funkce dochází k synchronizaci dvou nebo více paralelních úloh v konkrétním místě aplikace. Každá úloha, která dosáhne *Rendezvous*, čeká na určitý počet úloh, které jsou na tomto místě definovány. Výhodou *Rendezvous* je, že programátor může definovat počet úloh, na které se bude čekat. *Rendezvous* se dá použít jak v rámci jednoho virtuálního přístroje (dále jen VI), tak i v několika různých. [9]

Prostředky pro realizaci *Rendezvous* se nachází v synchronizační paletě >> Functions/Advanced/Synchronization/Rendezvous**,** viz. obr. 14.



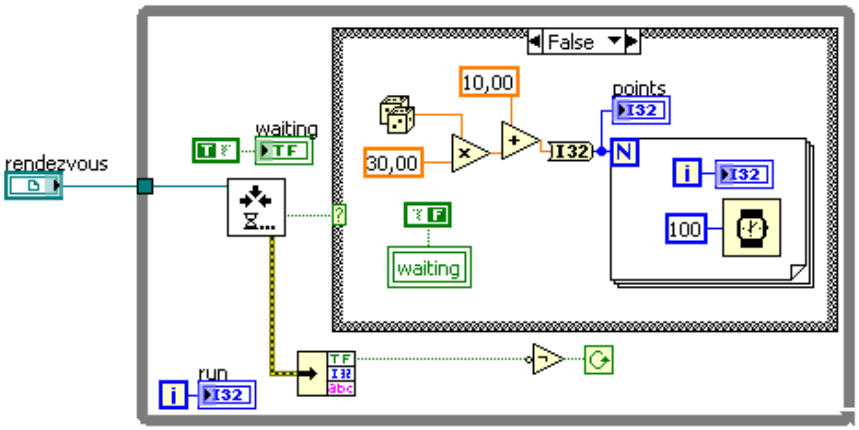

*Obr. 14 Paleta funkcí pro práci s rendezvous.*

Na obr. 15 je znázorněno nejzákladnější zavedení funkce *Rendezvous*. V tomto VI se zavádí proměnná, pro niž je umožněno definovat počet úloh, které se musí v tomto místě sejít a volají se *SubVI* obsahující jednotlivé úlohy.




*Obr. 15 Zavedeni Rendezvous. [10]*

V *SubVI* „User1“ zobrazeného na obr. 16 se pak čeká, dokud se v kritickém místě nesejde definovaný počet úloh a teprve pak se úloha vykoná.




*Obr. 16 Náhled do SubVI User1. [10]*

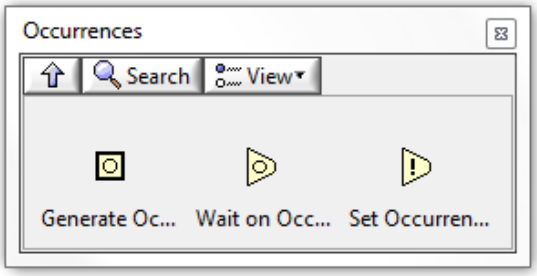
Konkrétní výčet funkcí palety *Rendezvous* [5]:

- **Create Rendezvous** zřizuje globální proměnnou s udáním počtu úloh, které se musí v místě *Rendezvous* shromáždit,
- **Wait at Rendezvous** čeká, dokud se nesejde definovaný počet úloh. Dobu čekání dáváme jako vstupní parametr,
- **Destroy Rendezvous** ruší proměnnou *Rendezvous*, všechny čekající úlohy vyhlásí ve výstupním parametru hodnotou TRUE a vyhlásí chybu,
- **Resize Rendezvous** mění se počet úloh, které se v místě schůzky musí sejít,
- **Get Rendezvous Status** zjišťuje stav proměnné *Rendezvous*,
- **Not a Rendezvous** zjišťuje, zda se jedná o *Rendezvous*.

### 5.2.2 Funkce Occurrences

Tato funkce se využívá zejména v případech práce v rámci jednoho VI nebo s částí blokového schématu čekajícího na jiné VI. *Occurrences* lze nahradit globální proměnnou, avšak její použití je v tomto případě vhodnější. Globální proměnná zatěžuje procesorový čas tím, že stále vyčítá její stav a čeká na jeho vhodnou změnu. Použitím *Occurrences* se do činnosti neuvádí čekající smyčka, tzn. nezatěžuje se procesorový čas, dokud nedojde k změně stavu u *Occurrences*. [5]

Prostředky pro realizaci *Occurrences* se nachází stejně jako předešlá funkci v synchronizační paletě >> Functions/Advanced/Synchronization/Occurrences, viz. obr. 17.



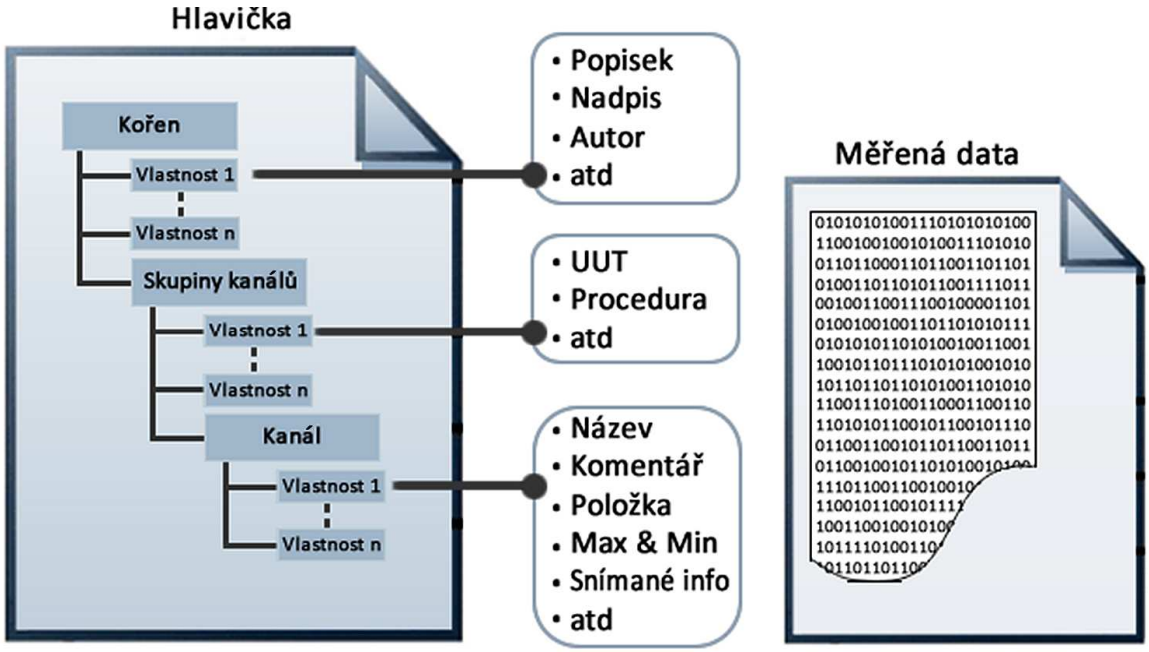

*Obr. 17 Paleta funkcí pro práci s Occurrences.*

## 5.3 Práce s daty pomocí TDMS

National Instruments vytvořil flexibilní datový model s názvem *TDM*, který je nativně dostupný prostřednictvím NI LabVIEW. Tento formát je přenosný i na další aplikace jako je Excel. Datový model *TDM* nabízí několik jedinečných výhod, jako je schopnost poskytnout uživateli širokou škálu nastavení, snadno připojit popisné informace. Model dat podporuje dva formáty, *TDM* a TDMStreaming (*TDMS*). *TDMS* je optimalizováno pro streamování a bezpečné ukládání na disk. *TDMS* model nabízí tři úrovně hierarchie, jak znázorňuje obr. 18 - *Kořen*, *Vlastnost* a *Kanál*. Každá úroveň akceptuje neomezený počet znaků – definované atributy. Na rozdíl od *TDM* souborů, které striktně vyžadují strukturování XML, *TDMS* binární soubory mají soubor s příponou *.tdms_Index, který obsahuje souhrnné informace o všech atributech a odkazy, čímž urychluje přístup k datům při čtení. *TDMS* je určeno pro rychlý a efektivní zápis dat na disk. [5]

Výčet výhod, které nabízí *TDMS* [5]:

- Rychlé streamování dat na disk ve strukturované podobě,
- Kategorizace údaje podle skupin,
- Použití real-time systémů,
- Snadné použití API,
- Usnadnění tvorby popisných údajů.



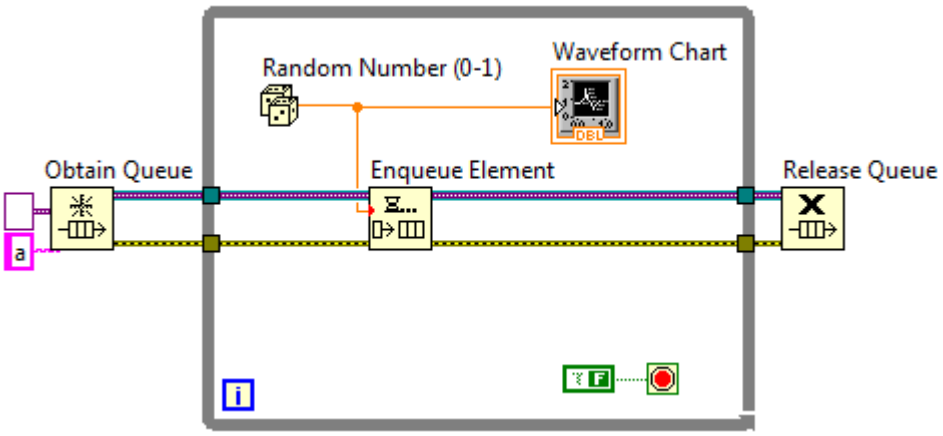

*Obr. 18 Hierarchie TDMS. [5]*

## 5.4 Zásobník FIFO – fronta

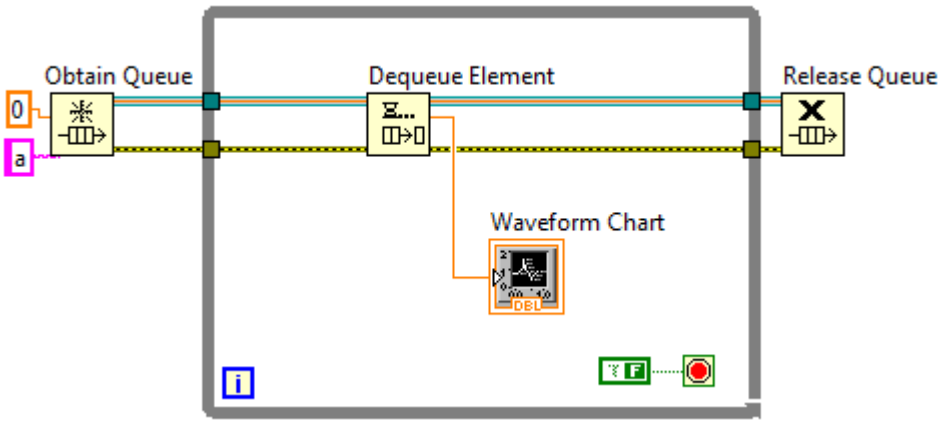
Synchronizační mechanizmus fronty se používá v situaci, kdy se předávají data ve stejném pořadí s využitím vyrovnávací paměti typu FIFO (First IN – First OUT). Zapsaná data zůstávají ve frontě, dokud nejsou vyčtena nebo explicitně odstraněna. [7]

Pro práci se zásobníkem dat je v NI LabVIEW implementován nástroj *Queue.* Tento nástroj umožňuje práci s daty tak, že je uchovává ve vyrovnávací paměti. Pomocí

této funkce je možné zabezpečit zpracování všech dat přicházejících v proudu (stream) a to i při kritických objemech dat a rychlostech jejich čtení. Pro názorný příklad práce se zásobníkem je na obr. 19 přidáváno do zásobníku náhodné číslo od nuly do jedničky, které je pak následně v jiném VI obr. 20 ze zásobníku vyjmuto a reprezentováno v grafu.




*Obr. 19 Práce s Queue – přidávání do zásobníku.*




*Obr. 20 Práce s Queue – vyjmutí ze zásobníku.*

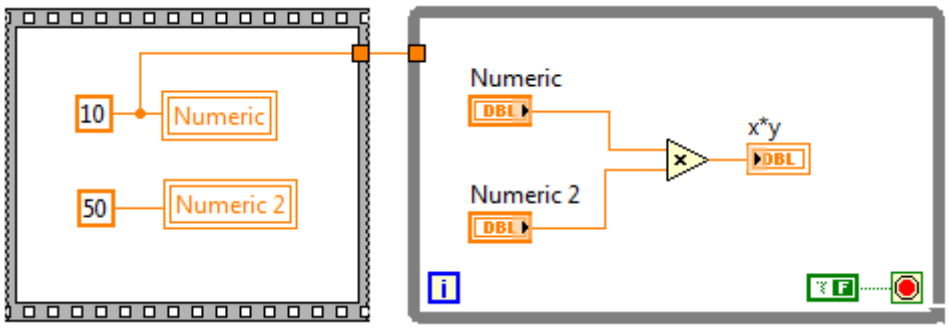
Pro práci s frontou jsou k dispozici následující funkce [5]:

- **Obtain Queue** vytvoří frontu s definovanou hloubkou,
- **Enqueue Element** vložení prvku na konec fronty,
- **Enqueue Element At Opposite End** vložení prvku na počátek fronty,
- **Dequeue Element** vyjmutí prvku z počátku fronty,
- **Preview Queue Element** vrátí prvek z počátku fronty,
- **Flush Queue** vyprázdnění fronty,
- **Release Queue** zruší frontu,
- **Get Queue Status** vrátí aktuální stav fronty.

## 5.5 Lokální proměnná

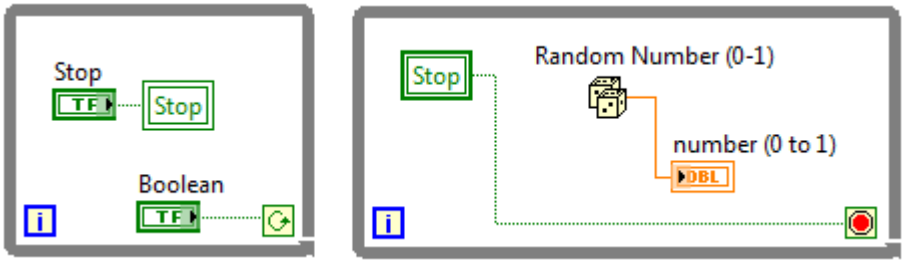
Každému objektu umístěnému na čelním panelu v programovém prostředí NI LabVIEW přísluší odpovídající koncový blok, pomocí něj vstupuje hodnota tohoto objektu do algoritmu. Koncový blok je buď v módu ovládacím, nebo indikačním. Současně je v paměti alokováno místo pro uložení odpovídající datové struktury, na které se ukládá aktuální hodnota objektu čelního panelu. Na toto místo v paměti lze tedy přistupovat pomocí koncového bloku. Pomocí *Local Variable* (dále jako lokální proměnné) lze zmnožit koncový blok příslušejícího objektu čelního panelu. Lokální proměnné se používají ve dvou standardních situacích [6]:

- Při potřebě programového zápisu do prvku v ovládacím módu, viz. obr. 21.




*Obr. 21 Možnost použití lokální proměnné typu double.*

- Při potřebě přístupu téže proměnné z více míst blokového diagramu, viz. obr. 22.




*Obr. 22 Možnost použití lokální proměnné typu boolean.*

Lokální proměnná má vlastnost, že buď do ní můžeme zapisovat, nebo z ní číst. Tato funkce se mění volbou módu.

## 5.6 Globální proměnná

*Global variable* (dále jako globální proměnná) se používá při potřebě sdílet proměnné několika virtuálními přístroji. Pomocí globální proměnné je umožněno vytvořit tzv. sdílenou datovou oblast, do které mohou přistupovat všechny virtuální přístroje daného projektu buď se zápisem, nebo pro čtení. Pro globální proměnnou stejně jako pro lokální platí, že do ní můžeme zapisovat, nebo z ní číst. [6]

Výhodou globální proměnné je, že může obsahovat více datových typů jiných objektů. Při její tvorbě si sám programátor definuje objekty, které má globální proměnná obsahovat a pak při práci s ní jen zvolí konkrétní objekt, se kterým chce pracovat v rámci

globální proměnné. Pokud programátor sváže objekty se sebou programově související do globální proměnné, dochází ke zpřehlednění kódu a omezení nadbytečného vytváření těchto proměnných.

## 5.7 Nástroje pro použití .NET kódu

Platforma *.NET Framework* je vytvořena společností Microsoft (MS) a je navržena k tomu, aby zjednodušila aplikační rozvoj ve vysoce distribuovaném prostředí internetu. *.NET Framework* podporuje mezinárodní standart CLI (Common Language Infrastructure) a existuje pro většinu 32 a 64-bitových operačních systémů bez potřeby překladu pro jiný systém nebo procesor. Pro běh .NET aplikací je potřeba mít nainstalovaný .NET Runtime. [3]

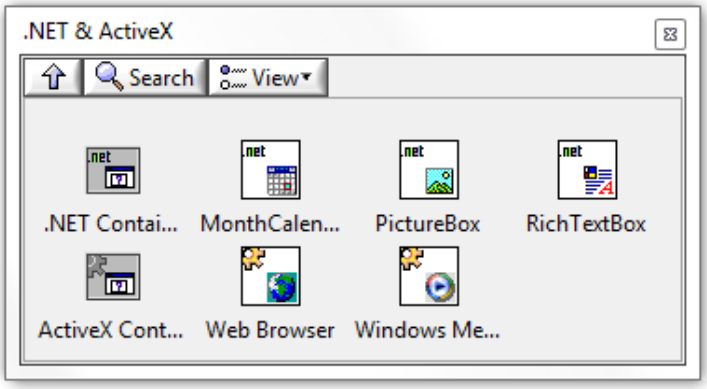
V této diplomové prácí byly využity funkce, které jsou implementované v prostředí NI LabVIEW 8.6. Pomocí těchto funkcí, jak bylo předesláno v kap. 3.2, je umožněna práce s již vytvořenými knihovnami *.dll. NI LabVIEW 8.6 nabízí širokou škálu prostředků pro práci s .NET. V rámci diplomové práce byly využity konkrétně funkce:

- *.NET Containers,*
- *Constructor node,*
- *Property node (.NET),*
- *Invoke node (.NET).*

Funkce *.NET Containers* patří k důležitým prostředkům pro reprezentace *windows forms* v NI LabVIEW a její využití přispělo značně ke tvorbě uživatelského interface, proto jí je věnována kapitola 5.7.1.

### 5.7.1 Funkce .NET Containers

Funkce *.NET Containers*, viz. obr. 23 se nachází na předním panelu (*Front panel*) NI LabVIEW v kontrolním menu >> .NET & ActiveX. Díky této funkci je umožněno pracovat s již vytvořenými knihovnami *.dll tak, že poskytují výhodu vložení *windows forms* na přední panel VI. Pomocí *.NET Containers* a vlastnosti scelect *.NET Control* je umožněno si načíst konkrétní *windows forms*. V programovém prostředí NI LabVIEW je umožněna práce s těmito formuláři a to tak, že data do nich můžeme vkládat, nebo z nich vyčítat a dále tak s načtenými daty pracovat.




*Obr. 23 Paleta funkcí .NET & ActiveX.*

**Část B - Praktická část**

# 6 REALIZACE NÁVRHU ŘÍDICÍHO SOFTWARE

Vytvořený software v prostředí NI LabVIEW 8.6 je rozdělen do dvou stěžejních kapitol. Kapitola 6.1 představuje konečný návrh uživatelského interface a vysvětluje principy a možnosti nastavení při spouštění dlouhodobých testů na otěr páteřních prvků. Kapitola 6.2 pojednává o programovém návrhu software k dlouhodobým experimentům, převodu uložených dat a software pro offline zobrazování dat. V předloženém software je možné vyčlenit tři logické celky, které jsou popsány v samostatných kapitolách; hlavní část software (kap. 6.1.1 a kap. 6.2.1), offline zobrazování dat (kap. 6.1.2 a kap. 6.2.2), konvertor dat (kap. 6.1.3 a kap. 6.2.3).

## 6.1 Uživatelské rozhraní

Jedná se o interaktivní grafické rozhraní (Graphical User Interface - GUI) určené koncovému uživateli tzv. čelní panel (*Front Panel*). Tento *Front Panel* obsahuje prvky pro ovládání a indikaci (grafy, tlačítka, LED indikátory, textové boxy, atd.). Takto vytvořený čelní panel je schopen uživatel ovládat pomocí myši nebo klávesnice. Využití funkce *.NET Containers*, viz. kap. 5.7.1 vytvořilo nemalou část grafické podoby uživatelského interface a poskytlo další výhody. Následující tři kapitoly se věnují konečnému návrhu vytvořeného čelního panelu.

### 6.1.1 Uživatelské rozhraní řídicího software

Po vzájemných konzultacích s pracovníky ÚMTMB byl vytvořen optimální návrh uživatelského rozhraní software pro řízení dlouhodobých experimentů zkoumajících vliv zatížení na velikost otěru páteřních prvků. Dojde-li ke spuštění software, znázorní se uživateli jako první okno s výběrem COM portu PC, ke kterému je zařízení připojeno. Následně se zobrazí uživatelské prostředí, které je členěno na pět záložek:

- *Settings*,
- *Motor 1*,
- *Motor 2*,
- *Motor 3*,
- *Reference position settings.*

Záložka *Settings* obsahuje hlavní ovládací a kontrolní prvky, viz. příloha 1. Uživatel zde má možnost nastavovat (editovat):

- Umístění a název souboru uchovávajícího data ve formátu *TDMS*, ukládaných za běhu experimentu,
- Počet požadovaných cyklů experimentu,
- Editační pole určené pro popis experimentu,
- Jméno uživatele provádějícího experiment,
- Rychlosti ukládání a online zobrazování dat v grafech (pomocí vodorovné posuvné lišty).

Dále jsou zde umístěny ovládací prvky pro:

- Odstartování experimentu,
- Pozastavení experimentu,
- Spuštění software určeného ke konverzi dat,
- Spuštění software pro offline zobrazení uložených dat,
- Zastavení experimentu,
- Zpřístupnění/znepřístupnění změn nastavení experimentu za chodu,
- Vypnutí/zapnutí ochrany proti možnému koliznímu nastavení pohybu a rychlosti pohonů,
- Nastavení mezních pohybů misek a rychlosti pohybu jednotlivých pohonů, viz. příloha 1. A,
- Smazáni dat zobrazených v grafech,
- Uzavření software (tlačítko „Exit“).

Na této záložce je také umístěno osm on-line aktualizovaných grafů zobrazujících aktuálně měřená data, viz. příloha 1. B. V grafech jsou vykresleny požadované závislosti měřených veličin, viz. kap. 6.2.1.2 - B. Jsou zde VI, kde se zobrazují aktuální číselné údaje o poloze a rychlosti pohybu pohonů. Při běhu experimentu se kontroluje možný výskyt chyb, které mohou vzniknout na zařízení. K jejich indikaci jsou zde umístěny (LED indikátory), viz. příloha 1. C.

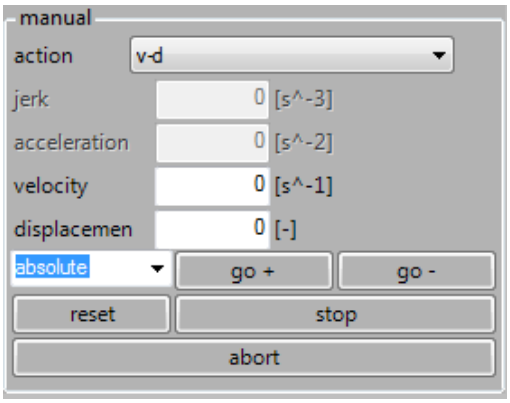
Záložky Motor 1, 2, 3 jsou vzhledově podobné. Každá ze záložek obsahuje data související s jedním ze tří pohonů. Obsahují *.NET Containers* zobrazující formuláře, které jsou vytvořeny a implementovány v podřízené vrstvě, viz. příloha 2. Číselné hodnoty těchto formulářů byly získány od Ing. Pavla Houšky, Ph.D. a nejsou obsahem této diplomové práce. Zmíněné záložky dále obsahují:

- Dvoupolohové tlačítko pro (de)aktivaci čtení momentu ze senzoru za pomocí funkce *DAQmx,*
- Tlačítka pro:
  - Zobrazení výchozích hodnot parametrů (získaných od Ing. Pavla Houšky, Ph.D.) do *.NET Containers* pro jednotlivé motory. Tyto parametry řízení jsou rozděleny do oblastí týkajících se nastavení:
    - Pohonu,
    - Převodovky,
    - IRC senzoru,
    - Teplotních, napěťových, momentových a napájecích omezení,
    - Parametrů PSD regulátoru pro polohové a rychlostní řízení.
    - (Podrobný výčet všech možných nastavení je nad rámec této práce).
  - Zapsání buďto načtených nebo uživatelem upravených hodnot v *.NET Containers* na zařízení. Pokud nedojde k zapsání parametrů řízení na testovací zařízeni, tak je tato informace o chybě nahlášena uživateli.

Záložka *Reference position settings* obsahuje *.NET Containers* formuláře, viz. obr. 24. Tyto formuláře umožňují uživateli manuální natočení jednotlivých pohonů do zvolené výchozí polohy. Pomocí tohoto formuláře lze nastavit počáteční polohy pohonů pro následující experiment. Postup:

- Do editačních polí *velocity, displacement* uživatel zadá rychlost pohybu pohonu a požadovaný úhel natočení, o který má být daná miska vychýlena,
- Stiskem tlačítka „go“ uživatel uvede do chodu konkrétní pohon nastavenou rychlostí a zajistí tak přírůstek natočení o zadaný úhel,
- Tlačítko „abort” slouží k odstranění původní, výchozí polohy,

- Stiskem tlačítka „reset“ je aktuální poloha misky nastavena jako nulová tzn. výchozí poloha.




*Obr. 24 .NET Containers pro nastavení výchozích poloh motoru.*

Informace o úhlu natočení misek, rychlosti atd. jsou obsaženy v *.NET Containers* (CntInputsAll_States), viz. příloha 3. A. Tyto průběžně aktualizované formuláře slouží uživateli pro kontrolu při nastavení výchozích poloh experimentu.

Posledním prvkem obsaženým v této záložce jsou dvě tlačítka určená k nulování momentu pro pohony namáhající páteřní prvek v ose flexe. Tato funkce byla implementována dodatečně, viz. kap. 6.2.1.2 - M.

### 6.1.2 Uživatelské rozhraní software pro offline zobrazování dat

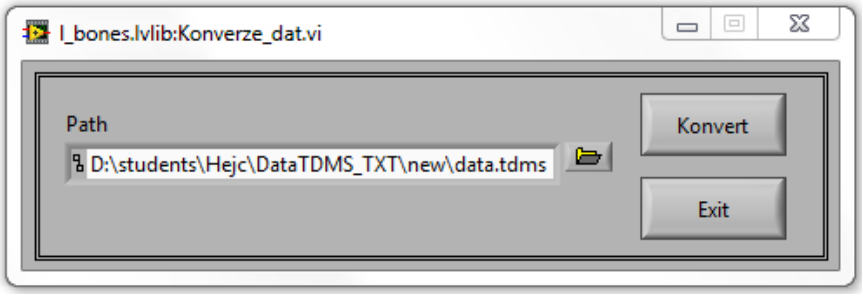
Uživatelské rozhraní je děleno pomocí *Tab control* na čtyři záložky. V hlavní záložce pojmenované *Measured data*, viz. příloha 4, uživatel zvolí adresář obsahující soubory s uloženými daty ve formátu *.tdms. Obsah adresáře se zobrazí jako seznam v komponentě *Listbox*. Data vybraného adresáře ze seznamu se zobrazí:

- V tabulce,
  - Úhly natočení misek,
  - Rychlosti pohonů,
  - Momenty,
  - Časový záznam právě provedeného cyklu,
  - Aktuální cyklus.
- V grafech jako závislosti,
  - Momentů na pohonech (1), (2) a (3), úhly natočení pohonů (1), (2) a (3) v závislosti na čase,
  - Momentů na pohonech (1), (2) a (3), úhly natočení pohonů (1), (2) a (3),
  - Momentů na pohonech (1, 2, 3) v závislosti na čase,
  - Úhly natočení pohonů (1, 2, 3) v závislosti na čase.
- V informačních vícepoložkových polích.
  - Popis experimentu,
  - Jméno uživatele provádějícího experiment,
  - Počet požadovaných zátěžných cyklů,
  - Časový záznam, kdy byl experiment zahájen/pozastaven,
  - Časový záznam ukončení experimentu.

Grafy obsažené v tomto software umožňují snadnou modifikaci zobrazení, viz. příloha 5. Každý z grafů můžeme lehce přibližovat, soustředit se jen na vybraná data a snadno se v něm pohybovat. Záložka *Measured data* obsahuje tlačítko „TDMS Viewer“, které vyvolá stejnojmennou funkci implementovanou v NI LabVIEW 8.6. *TDMS Viewer* je funkce, která umožňuje základní náhled do souboru *.tdms. Hierarchie dat, která jsou zobrazena pomocí této funkce, je tvořena pomocí pojmenování konkrétní skupiny dat vytvořených při ukládání. Důvodem využití této funkce bylo poskytnutí možnosti porovnání zobrazení dat vyvíjenou aplikací se zobrazením výchozím (nezpracovaným), což se ukázalo jako užitečné především během vývoje aplikace. Nyní je zde funkce ponechána pro poskytnutí alternativního náhledu na naměřená data.

### 6.1.3 Uživatelské rozhraní software pro konverzi dat

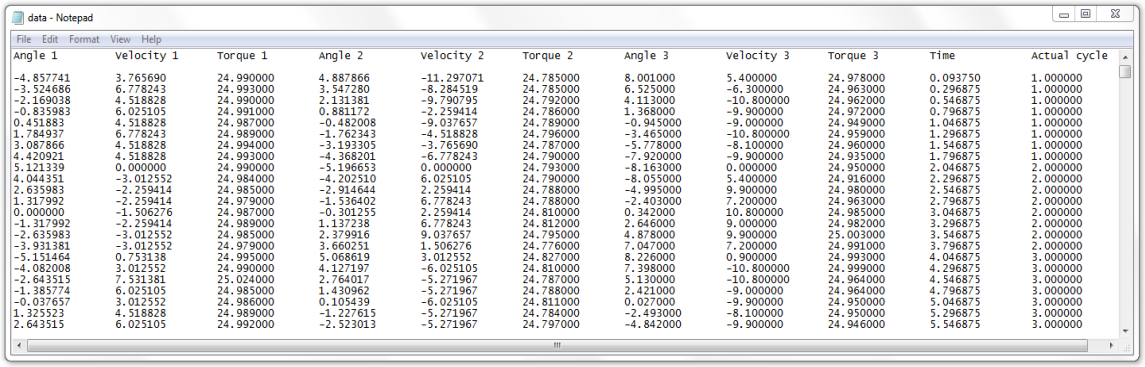
Na základě požadavků, viz. kap. 4.3 byl vytvořen software, který umožňuje převést uložená data ve formátu *.tdms do univerzálního formátu *.txt, a to z důvodu případného dodatečného zpracování dat pomocí jiných programů jako např. Matlab. Hlavním důvodem ukládání dat ve formátu *TDMS* je využití této funkce implementované v prostředí NI LabVIEW 8.6 pro bezpečné a rychlé ukládání velkého množství dat, viz. kap. 5.3. Uživatel si zvolí jaký soubor *.tdms chce převádět, viz. obr. 25. Spuštěním programu (tlačítko „Konvert“) si volí, na jaké místo na disku se data uloží v požadovaném formátu *.txt.




*Obr. 25 Interface software pro konverzi dat.*

Při konverzi dojde k automatickému uložení dat do dvou souborů:

- Soubor obsahující měřená data, viz. obr. 26. Obsahuje jedenáct sloupců tak, že v prvních třech jsou udány úhly natočení misek, rychlosti a průběh momentu vztahující se k prvnímu z pohonů. Ve stejném pořadí jsou zobrazeny zbylé dva pohony. Obsah posledních dvou sloupců je časový okamžik, ve kterém došlo k zaznamenání hodnot jednotlivých veličin a aktuálně probíhajícího cyklu. Každý ze sloupců je opatřen názvem vztahujícím se k datům obsaženým ve sloupci, viz. obr. 26. Jednotlivé sloupce jsou odděleny tabulátorem.


data - Notepad

File Edit Format View Help

| Angle 1 | Velocity 1 | Torque 1 | Angle 2 | Velocity 2 | Torque 2 | Angle 3 | Velocity 3 | Torque 3 | Time | Actual cycle |
|---|---|---|---|---|---|---|---|---|---|---|
| -4.857741 | 3.765690 | 24.990000 | 4.887866 | -11.297071 | 24.785000 | 8.001000 | 5.400000 | 24.978000 | 0.093750 | 1.000000 |
| -3.524686 | 6.778243 | 24.993000 | 3.547280 | -8.284519 | 24.785000 | 6.525000 | -6.300000 | 24.963000 | 0.296875 | 1.000000 |
| -2.169038 | 4.518828 | 24.990000 | 2.131381 | -9.790795 | 24.792000 | 4.113000 | -10.800000 | 24.962000 | 0.546875 | 1.000000 |
| -0.835983 | 6.025105 | 24.991000 | 0.881172 | -2.259414 | 24.786000 | 1.368000 | -9.900000 | 24.972000 | 0.796875 | 1.000000 |
| 0.451883 | 4.518828 | 24.987000 | -0.482008 | -9.037657 | 24.789000 | -0.945000 | -9.000000 | 24.949000 | 1.046875 | 1.000000 |
| 1.784937 | 6.778243 | 24.989000 | -1.762343 | -4.518828 | 24.796000 | -3.465000 | -10.800000 | 24.959000 | 1.296875 | 1.000000 |
| 3.087866 | 4.518828 | 24.994000 | -3.193305 | -3.765690 | 24.787000 | -5.778000 | -8.100000 | 24.960000 | 1.546875 | 1.000000 |
| 4.420921 | 4.518828 | 24.993000 | -4.368201 | -6.778243 | 24.790000 | -7.920000 | -9.900000 | 24.935000 | 1.796875 | 1.000000 |
| 5.121339 | 0.000000 | 24.990000 | -5.196653 | 0.000000 | 24.793000 | -8.163000 | 0.000000 | 24.950000 | 2.046875 | 2.000000 |
| 4.044351 | -3.012552 | 24.984000 | -4.202510 | 6.025105 | 24.790000 | -8.055000 | 5.400000 | 24.916000 | 2.296875 | 2.000000 |
| 2.635983 | -2.259414 | 24.985000 | -2.914644 | 2.259414 | 24.788000 | -4.995000 | 9.900000 | 24.980000 | 2.546875 | 2.000000 |
| 1.317992 | -2.259414 | 24.979000 | -1.536402 | 6.778243 | 24.788000 | -2.403000 | 7.200000 | 24.963000 | 2.796875 | 2.000000 |
| 0.000000 | -1.506276 | 24.987000 | -0.301255 | 2.259414 | 24.810000 | 0.342000 | 10.800000 | 24.985000 | 3.046875 | 2.000000 |
| -1.317992 | -2.259414 | 24.989000 | 1.137238 | 6.778243 | 24.812000 | 2.646000 | 9.000000 | 24.982000 | 3.296875 | 2.000000 |
| -2.635983 | -3.012552 | 24.985000 | 2.379916 | 9.037657 | 24.795000 | 4.878000 | 9.900000 | 25.003000 | 3.546875 | 2.000000 |
| -3.931381 | -3.012552 | 24.979000 | 3.660251 | 1.506276 | 24.776000 | 7.047000 | 7.200000 | 24.991000 | 3.796875 | 2.000000 |
| -5.151464 | 0.753138 | 24.995000 | 5.068619 | 3.012552 | 24.827000 | 8.226000 | 0.900000 | 24.993000 | 4.046875 | 3.000000 |
| -4.082008 | 3.012552 | 24.990000 | 4.127197 | -6.025105 | 24.810000 | 7.398000 | -10.800000 | 24.999000 | 4.296875 | 3.000000 |
| -2.643515 | 7.531381 | 25.024000 | 2.764017 | -5.271967 | 24.787000 | 5.130000 | -10.800000 | 24.964000 | 4.546875 | 3.000000 |
| -1.385774 | 6.025105 | 24.985000 | 1.430962 | -5.271967 | 24.788000 | 2.421000 | -9.000000 | 24.964000 | 4.796875 | 3.000000 |
| -0.037657 | 3.012552 | 24.986000 | 0.105439 | -6.025105 | 24.811000 | 0.027000 | -9.900000 | 24.950000 | 5.046875 | 3.000000 |
| 1.325523 | 4.518828 | 24.989000 | -1.227615 | -5.271967 | 24.784000 | -2.493000 | -8.100000 | 24.950000 | 5.296875 | 3.000000 |
| 2.643515 | 6.025105 | 24.992000 | -2.523013 | -5.271967 | 24.797000 | -4.842000 | -9.900000 | 24.946000 | 5.546875 | 3.000000 |

*Obr. 26 Naměřená data, která jsou převedena do formátu *.txt.*

- Za název druhého souboru se automaticky z důvodu rozlišení oproti prvnímu přidá název „_info“ a je uložen automaticky zároveň s prvním souborem. Tento soubor slouží jako informativní a obsahuje data potřebná k opakování testu. Dojde-li při průběhu experimentu k pauze nebo ke změnám nastavení za běhu, pak dojde v datech k oddělení údajů znakem „ / “, jak je patrné z přílohy 6. Struktura tohoto souboru obsahuje:
  - Informace obecné,
    - Autor,
    - Popis testu,
    - Počet požadovaných cyklů,
    - Časový záznam spuštění/pozastavení experimentu,
    - Časový záznam ukončení experimentu.
  - Parametry nastavení pohybu u jednotlivých pohonů,
    - Krajní úhel pohybu misek,
    - Zadaná rychlost pohonu.
  - Nastavení obsažené v *.NET Containers* pro jednotlivé pohony.
    - Podrobný výčet je nad rámec obsahu této diplomové práce.

## 6.2 Blokové schéma

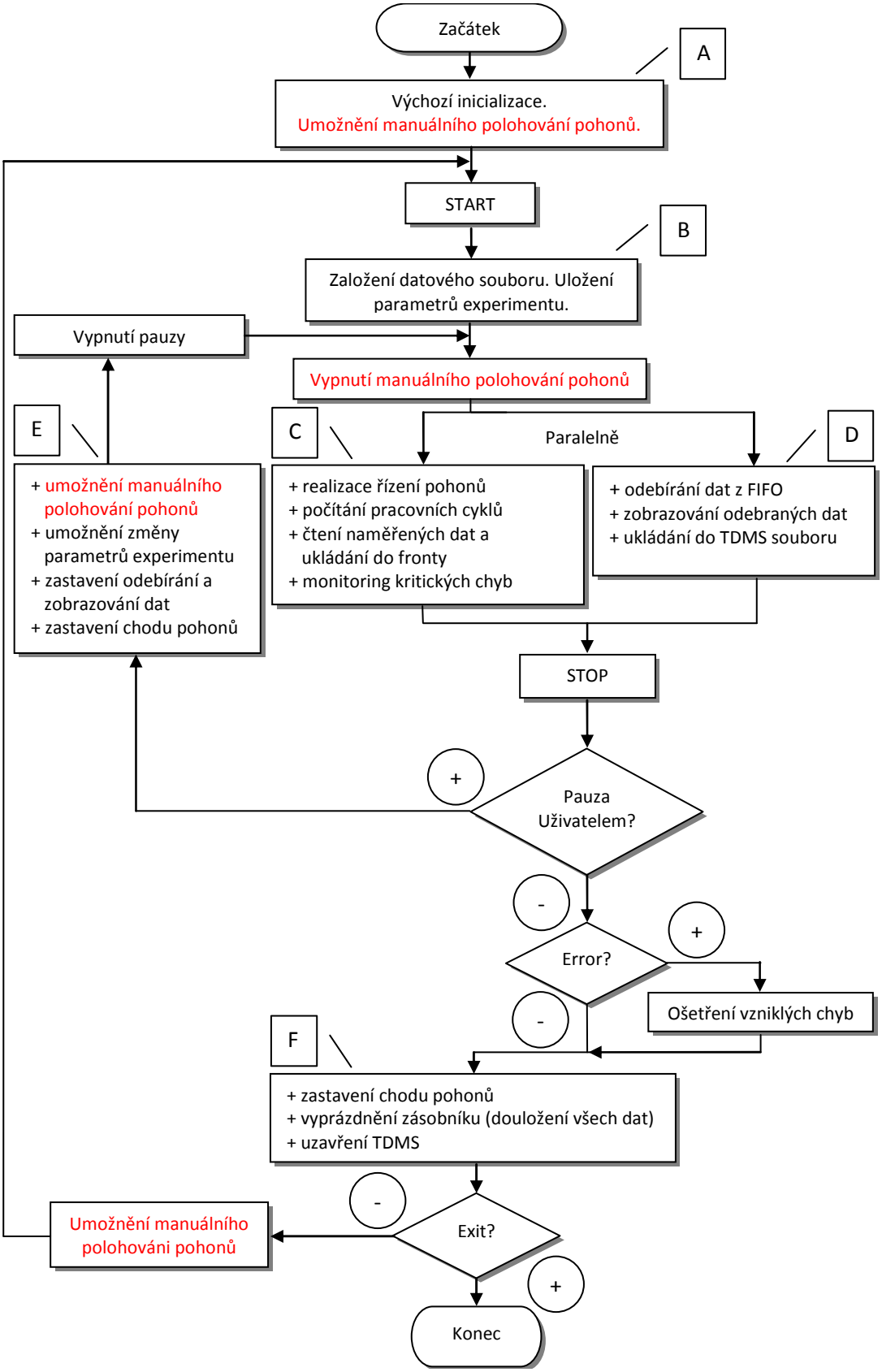
Blokovým schématem je myšlen programátorem vytvořený návrh zdrojového kódu každého software vytvořeného v prostředí NI LabVIEW pomocí grafického programování. Takový návrh zdrojového kódu se v NI LabVIEW nazývá *Block Diagram*. Toto blokové schéma je vytvořeno ikonami reprezentujícími ovládací a indikační prvky čelního panelu, viz. kap. 6.1. Blokový diagram je zdrojovou podobou každé aplikace vytvořené v prostředí NI LabVIEW.

Virtuální přístroj má hierarchickou modulární strukturu. Lze jej používat jako celý program nebo jeho jednotlivé podprogramy, které se nazývají podřízenými virtuálními přístroji tzv. *SubVI*. Díky těmto *SubVI* je software tvořen modulárně. Moduly jsou děleny na jednotlivé úlohy. Pod pojmem *SubVI* , jež je na nejnižší úrovni grafického programování, si také můžeme představit proceduru či funkci z klasického programování. Logickou tvorbou a spojováním vytvořených *SubVI* do VI je budována celá aplikace. Díky možnosti vyzkoušet funkci každého dílčího virtuálního přístroje nezávisle na jiných a díky bohaté škále ladících prostředků, je ladění aplikace velmi usnadněno.

### 6.2.1 Blokové schéma řídicího software

Toto blokové schéma obsahuje 78 *SubVI*, které jsou vytvořené pro řešení dílčích problémů. Pro snazší popis blokového schématu předloženého software byl sestaven zjednodušený vývojový diagram, viz. obr. 27. Vzhledem k rozsahu popisovaného software, zde bylo nutné některé části kódu nebrat v potaz. Vynechané části jsou však detailně popsány v textu. Dále je nutné poznamenat, že aplikace není řízena tokem dat, ale vyskytujícími se událostmi, proto je nutné uvedený vývojový diagram chápat pouze orientačně.

Na obr. 27 je znázorněno schéma základní myšlenky chodu software pro dlouhodobé testy na páteřních prvcích. Jak je vidět na obrázku, bezprostředně po začátku chodu software dojde k inicializaci a uživateli je umožněno manuální polohování pohonů, viz. blok A. Pokud uživatel odstartuje experiment, dojde k uložení nastavených parametrů řízení a zároveň je znemožněno uživateli manuálně polohovat, viz. blok B. Bloky C a D jsou prováděny paralelně tak, že v jejich rámci jsou řešeny všechny hlavní operace, viz. obr. 27. Mohou nastat tři případy zastavení chodu zařízení, při kterých dojde k přerušení paralelního běhu zmíněných bloků C a D. Prvním, ve schématu uvedeným případem, je pozastavení prostřednictvím pauzy, tj. dočasné zastavení, při kterém dojde k zastavení chodu pohonů i odebírání dat ze zařízení. Uživateli jsou následně umožněny změny nastavení parametrů a funkce manuálního polohování, viz. blok E. Po vypnutí pauzy dojde k opětovnému spuštění, viz. schéma. Druhým případem zastavení je reakce na vzniklou chybu na zařízení, při které se samočinně provede obsah bloku F. Tento blok se provede také v případě, kdy uživatel ukončuje experiment. Uživateli je následně umožněno manuální polohování a vytvoření nového experimentu.




*Obr. 27 Schéma základního běhu software pro dlouhodobé testování.*

Popis blokového schématu programu je rozdělen do čtyř oblastí:

- Inicializace použitých prvků,
- Reakce na události,
- Realizace chodu pohonů,
- Ukončení experimentu.

Tyto oblasti jsou mezi sebou propojeny a navzájem synchronizovány tak, aby byl dodržen plynulý chod pohonů a bylo zamezeno možné ztrátě dat, která jsou výstupem každého experimentu, při výskytu kritických chyb za běhu software, viz. následující text.

#### 6.2.1.1 Inicializace

Inicializační část, která je ve vývojovém diagramu uvedena jako blok A, je znázorněna blokovým schématem, viz. příloha 7. Jak je vidět v příloze, je postupně dělena pomocí programové struktury *Sequence*. Tato programová struktura nemá ekvivalent v textově orientovaných programových jazycích. V grafickém programování *Sequence* řídí posloupnost provádění příkazů a tok dat. Jednotlivé rámce jsou prováděny postupně v pořadí podle čísla v záhlaví této programové struktury *Stacked Sequence* nebo po sobě jdoucích rámců *Flat Sequence*. V inicializační části programu se využívá mimo jiné lokálních i globálních proměnných, viz. kap. 5.5 a 5.6. Lokální proměnné jsou použity převážně z důvodu neopakujících se volání stejných částí programu, což má za následek zvýšení přehlednosti a zjednodušení programu. Inicializační část je řešena pomocí čtyř *Flat Sequence* a jedné *Stacked Sequence*, která obsahuje osm rámců. Důležitým prvkem inicializační části je vytvoření *NI DAQmx Start Task* (popis funkce viz. kap. 5.1), která se nachází v pátém rámci *Stacked Sequence*. Tuto funkci zavádíme za účelem čtení dat ze zařízení za pomoci karty NI PCI-6220. V inicializační části pomocí *Sequence* řešíme celou řadu inicializací používaných prvků, mezi něž patří:

- Ovládací, indikační a uživatelské prvky (tlačítka, LED indikátory, vypínače/přepínače, grafy, atd.),
- Zavedení všech použitých *.NET Containers* (význam funkce viz. kap. 5.7.1),
- Nastavení parametrů pro zvolený typ prototypu přístroje, viz. kap. 1 k testování otěru páteřních prvků. Pro konstrukčně zdokonalené testovací zařízení používané v rámci této diplomové práce jsou zadány číselné hodnoty, které byly získány od tvůrce prostřední vrstvy Ing. Pavla Houšky, Ph.D. ,
- Omezení horní a spodní hranice uživatelem nastavitelných úhlů pohybu misek a rychlosti pohonu tak, aby nedošlo k možné kolizi,
- Zavedení stěžejní funkce *Rendezvous* pro potřebu synchronizace (význam funkce viz. kap. 5.2.1) řízení chodu smyček v programu.

Inicializační část je logicky řešena pomoci sedmnácti *SubVI*, ve kterých je mimo jiné prováděno:

- Vytvoření instancí třídy *UniversalMotorControler*, která umožňuje práci s oběma typy pohonu, viz. kap. 1 (řízení, indikace stavu, atd.),
- Připojení k PC – číslo COM portu a jeho přenosovou rychlost,
- Uvedení tlačítek do výchozích poloh.

#### 6.2.1.2 Vzniklé události

Vzniklé události jsou odchytávány pomocí smyčky *Event Structure*, která umožňuje provádět příkazy na základě výskytu událostí. Z důvodu nutnosti stálého opakování *Event Structure* je tato smyčka obsažena ve smyčce *While Loop*. *While Loop* je cyklus, jehož opakování je dáno testovanou podmínkou tj. logickým stavem tlačítka „Exit“ umístěným na předním panelu, viz. kap. 6.1.1. Dokud testovaná podmínka nezmění svůj stav tak, aby došlo k ukončení této smyčky, tak vše, co se nachází uvnitř této struktury, je cyklicky opakováno. *Event Structure* obsahuje 41 *SubVI,* pomocí nichž řeší úkoly spjaté s výskytem 25 odchytávaných událostí:

A ) Zahájení experimentu,
B ) Ukládání zaznamenávaných dat, jejich číselného zobrazování a on-line zobrazování v osmi grafech, indikace výskytu chyb,
C ) Pozastavení experimentu,
D ) Zastavení experimentu,
E ) Zapsání hodnot parametrů řízení na zařízení,
F ) Načtení základních parametrů pro řízení,
G ) Smazání dat vykreslených v grafech,
H ) Spuštění software pro konverzi uložených dat,
I ) Spuštění software pro offline zobrazování uložených dat,
J ) Vypnutí ochrany maximálních úhlů pohybu misek a přípustných rychlostí,
K ) Zapnutí ochrany maximálních úhlů pohybu misek a přípustných rychlostí,
L ) Umožnění uživateli měnit za chodu experimentu, tzn. bez pozastavení, parametry pro pohyb pohonů a rychlosti ukládáni, zobrazování dat,
M ) Nastavení nulového momentu.

**A) Zahájení experimentu**

Experiment uživatel zahájí stisknutím tlačítka „Start“ umístěným na předním panelu VI. Tímto je vyvolána událost *Button Start,* viz. příloha 8. Následek této události je také z důvodu přehlednosti prezentován ve schématu, viz obr. 27 - B. V rámci této události jsou provedeny následující úkony (výběr těch významnějších):

- Uvedou se do stavu neaktivity vybraná tlačítka a ovládací prvky umístěné na předním panelu.
- Založí se datový soubor a to pouze v případě, že se jedná o první spuštění tzn. záleží na stavu lokální proměnné „pause“, tak se v rámci této události vytvoří soubor (pojmenovaný uživatelem) *.tdms a do něj se automaticky uloží veškerá data obsažená v *.NET Containers* umístěná na předním panelu, viz. kap. 6.1.1. V druhém případě, tj. opětovném spuštění po pauze chodu, se tento úkon neprovádí.
- Provede se smazání zobrazených dat v grafech.
- Inicializuje se lokální proměnná „IsRuning“ typu boolean. Tato lokální proměnná má velký vliv na chod dalších částí programu konkrétně:
    - Odstartuje *Case Structure* - přepínač, která obecně řeší situaci, kdy je nutno na základě nějaké podmínky algoritmus větvit do dvou nebo více směrů. Význam použití *Case Structure* je popsán v následující kap. 6.2.1.3 a provádí realizaci chodu motorů. Na obr. 27 je tato část reprezentována pod písmenem C.

- V události *TimeOut* uvádí do chodu za použití *Case Structure* proces ukládání a vykreslování dat v grafech - popis události je obsažen v bodu B) Práce se zaznamenávanými daty.

**B) Práce se zaznamenávanými daty**

Patří mezi nejobsáhlejší části *Event Structure* a je řešena výskytem události *TimeOut*, viz. příloha 9. Cyklické opakování této události je dle požadavků uživatele volitelné v rozsahu 250 – 2500 milisekund, viz. kap. 4.1. Tento interval udává, jak často jsou čtena měřená data. *Case Structure* obsažena v této smyčce obsahuje celý algoritmus ukládání a online zobrazování zaznamenávaných dat. Začlenění této události do celkového chodu software je znázorněno na obr. 27 - D. *Case Structure* je řízen pomocí logické proměnné „IsRuning“ obsažené v lokální proměnné, jak bylo předesláno v předchozím textu. Zaznamenaná data jsou vyčítána ze zásobníku FIFO (obecný popis funkce, viz. kap. 5.4) a dále jsou zpracovávána k:

- Ukládání dat do souboru na disk ve formátu *.tdms,
- On-line zobrazování ukládaných dat v číselné podobě na předním panelu,
- On-line vykreslování dat pomocí osmi grafů jako závislosti:
  - Momentů na pohonech (1), (2) a (3), úhly natočení pohonů (1), (2) a (3) v závislosti na čase,
  - Momentů na pohonech (1), (2) a (3), úhly natočení pohonů (1), (2) a (3),
  - Momentů na pohonech (1, 2, 3) v závislosti na čase,
  - Úhly natočení pohonů (1, 2, 3) v závislosti na čase.
- Inicializaci možných výskytů chyb.

Pomocí vytvořeného VI se k pravidelně ukládaným datům uloží i údaje o prováděném experimentu, který si vyžádali pracovníci ÚMTMB. Toto VI je schopno ukládat požadované údaje jen v případech, pokud nastane:

- Stisknutí tlačítka „Start“ umístěného na předním panelu,
- Za běhu experimentu, dojde-li k pozastavení chodu pohonu za pomocí tlačítka „Pause“,
- Za běhu experimentu, v případě, kdy uživatel zvolí možnost měnit nastavení pohybu misek a rychlostí pohonů bez toho, aniž by došlo k pozastavení chodu pohonů (ovládací prvek „Lock/unlock change data“).

Ve zmíněných událostech dojde k uložení údajů v prováděném experimentu, aniž by došlo k přepsání předchozího záznamu. Ukládané údaje jsou:

- Časový záznam, kdy experiment začal nebo byl pozastaven (provedena změna nastavení),
- Jméno uživatele provádějícího experiment,
- Popis experimentu vytvořený uživatelem,
- Konečný počet cyklů, které má zařízení vykonat,
- Maximální natočení a rychlosti chodu jednotlivých pohonů, které si uživatel zvolil.

**C) Pozastavení experimentu**

Při stisknutí tlačítka „Pause“ umístěném na předním panelu dojde k vyvolání události, která má za následek pozastavení chodu motorů. Názorná ukázka významu a začlenění této funkce je schematicky znázorněna na obr. 27. Tato funkce byla vyžádána uživatelem, viz. kap. 4.1. V této události se inicializuje stav lokální proměnné „pause“. Stav této proměnné se dále využívá v programové části k:

- Realizací chodu pohonů, kde se na základě stavu provedou patřičné operace. Podrobný obsah využití logického stavu této proměnné je obsažen v popisu této programové části, viz. kap. 6.2.1.3.
- Zahájení experimentu, kdy za použití stavu této logické proměnné dojde/nedojde k založení souboru *.tdms a následnému uložení dat z *.NET Containers*, viz. A) Zahájení experimentu.

V rámci této události se provede zpřístupnění tlačítek a editačních polí. A tím je umožněno uživateli editovat nastavení experimentu a je pozastaveno odebírání dat ze zařízení zařazení, viz. obr. 27 – E. Obecně při použití této funkce nesmí dojít k uzavření souboru, do kterého jsou data ukládána, jelikož při uvedení přístroje do opětovného chodu je na předchozí data navázáno.

**D) Zastavení experimentu**

Událost zastavení experimentu je vyvolána za pomocí tlačítka „Stop“ umístěném na předním panelu. Logické zařazení této události do chodu celého software je znázorněno na obr. 27 – F. Uživateli je umožněno mimo jiné vytvořit název pro nový adresář pro uložení nových dat a nastavit parametry nového experimentu. Tato událost obsluhuje globální proměnnou (popis obecných vlastností globální proměnné, viz. kap. 5.6), jejíž hodnota se nastaví na „zastavit“. Díky této globální proměnné zastavíme hlavní smyčku chodu pohonů, viz. kap. 6.2.1.3**.**

**E) Zapsání hodnot parametrů řízení na zařízení**

Zapsání hodnot parametrů řízení je vyvolána události *OnClick* tlačítkem „Set parameters“. Pomocí vytvořených *SubVI* je uživatel schopen pro jednotlivé motory zapsat např. hodnoty pro:

- Řízení (otáčky za minutu, převodový poměr),
- IRC senzory (čas vzorkování, rozlišení),
- Teplotní senzory (případné omezení, teplotní limit),
- Napájení (napájecí limit),
- Kalibrace (vyrovnávání, reference),
- Ad. (ucelený výčet přesahuje stanovený rozsah této práce).

Tato funkce je řešena v části *Event Structure*. Nedojde-li k zapsání parametrů řízení, je tento proces třikrát opakován pro každý z pohonů, následně v případě neúspěchu program nahlásí poruchu v připojení zařízení.

**F) Načtení základních parametrů pro řízení**

Uživateli je umožněno pomocí tlačítka „Set Default “ zavolat událost, která do *.NET Containers* načte doporučené základní parametry řízení pro optimální chod zařízení. Načtení je řešeno pomocí funkcí v NI LabVIEW umožňující přístup k vlastnostem jednotlivých položek obsažených v podřízené vrstvě. Pomocí přístupu do těchto vlastností pro každou z položek reprezentovaných v *.NET Containers* došlo ke striktnímu zadání parametrů obsažených v položce reprezentované v *.NET Containers*. Díky velkému počtu těchto položek došlo k vytvoření rozsáhlého *SubVI*, které tyto základní parametry řízení zobrazí v položkách umístěných v *.NET Containers*. Pro ukázku bylo zvoleno zapsání základních parametrů pro nastavení parametrů pohonu a převodovky, viz. příloha 10.

### G) Smazání dat vykreslených v grafech

Dalším požadavkem uživatele je možnost mazání dat, která jsou vykreslena v grafech. Pro tuto funkci je zde událost vyvolána na *Value Change* tlačítka „Clear Graphs“, které je umístěno na předním panelu. V této události je umístěno VI, které pomocí vlastností grafu (ValSgnl) načte do všech grafů umístěných na front panelu prázdné hodnoty.

### H) Spuštění software pro konverzi uložených dat

Jak bylo zmíněno v kap. 4, jedním z požadavků bylo sjednocení vytvořených software. Prostředí NI LabVIEW 8.6 umožňuje volání samostatně vytvořených aplikací umístěných ve stejném projektu. V rámci této diplomové práce byly vytvořeny tři software, viz. kap. 4:

- Řídicí software,
- Software pro offline zobrazování uložených dat ve formátu *.tdms,
- Software pro převod *.tdms do univerzálního formátu *.txt.

Pro každý ze jmenovaných software je vytvořeno samostatné *.VI. Pomocí události (*Value Change* na tlačítko „Open Convertor“) a v ní obsažené funkce pro volání *.VI je umožněno spouštět software pro konverzi dat. Software pro konverzi dat je uživateli zpřístupněn spouštět i za chodu experimentu a to tak, že hlavní okno řídicího software se uvede pouze do stavu nečinnosti (za stálého provozu řídicího software) a okno pro konverzi dat vystoupne nad něj a uživateli je dovoleno s ním pracovat. Blokové schéma software pro konverzi dat je popsáno v kapitole 6.2.3.

### I) Spuštění software pro offline zobrazování uložených dat

Událost pro spuštění software pro offline zobrazování uložených dat pracuje obdobným způsobem jako předešlá událost pro tlačítko „Open Convertor“ s tím rozdílem, že ji uvádíme do činnosti *Value Change* tlačítkem „Open DataView“ umístěném na předním panelu. Blokové schéma pro software umožňující zobrazování uložených dat ve formátu *.tdms je popsáno v kap. 6.2.2.

### J) Vypnutí ochrany maximálních úhlů pohybu misek a přípustných rychlostí

Událost *Mouse Down* pro vypnutí ochrany se provede stisknutím tlačítka „Turn off protection“. Výchozí stav je takový, že ochrana krajních pohybů misek a rychlosti jednotlivých pohonů je zapnutá. Pokud by uživatel chtěl z jakéhokoli důvodu tuto ochranu dočasně vypnout, je pomocí vlastností (maximum, minimum) editačních prvků umístěných na předním panelu umožněno těmto prvkům nastavit neomezující hodnoty (INF). Událost v sobě obsahuje funkci *Dispaly Message to User*, pomocí které je uživateli oznámeno varování.

### K) Zapnutí ochrany maximálních úhlů pohybu misek a přípustných rychlostí

Zpětné aktivování ochrany je využito událostí *Mouse Down* na tlačítku „Turn on protection“. Opět je využito vlastností editačních polí (maximum, minimum), ale s tím rozdílem, že jsou do nich alokovány testy ověřené krajní hodnoty. Tyto hodnoty uživateli zajistí ochranu proti možné kolizi chybným nastavením.

### L) Umožnění změn parametrů experimentu během jeho chodu

Jedním z mnoha požadavků je také umožnit uživateli měnit parametry experimentu, viz. kap. 4.1 a to za chodu experimentu. Proto byla vytvořena událost *Value Change* na

kolébkové tlačítko „Lock/Unlock change data“ umístěném na předním panelu. Událost obsahuje dvě *Case Structure*. Pomocí vlastností tlačítka „Lock/Unlock change data“ je vybráno jedno z programových oken prvního ze dvou *Case Structure.* Pokud je tlačítko v aktivním stavu, je uživateli umožněno měnit parametry experimentu, viz. kap. 4.1. V případě, kdy uživatel změní stav tlačítka oproti původnímu stavu, dojde ke zrušení aktivního stavu doposud uživateli zpřístupněných editačních polí. Druhý ze zmíněných *Case Structure* obsahuje globální proměnnou „saveTestSettingMotor“, pomocí níž se odkazuje do části *TimeOut*, kde se do *.tdms souboru uloží údaje o změně parametrů experimentu, takže nedojde k nahrazení původních.

**M) Nastavení nulového momentu**

Toto nastavení bylo dodatečným požadavkem ze strany uživatele. Požadavek se týká situace, kdy je testovaný páteřní prvek předzatížen a uživatel si přeje tento stav prohlásit za stav, v němž působí nulový moment. Událost *MouseDown* na tlačítka „Vynuluj moment“ umístěném na předním panelu, nastavuje globální proměnnou „zerroTorgue“. Tato proměnná je použita v části realizace chodu motorů v *SubVI* „Synchronization“, kde je od zaznamenané hodnoty změřeného momentu odečtena hodnota proměnné „zerroTorgue“, tj. všechny změřené hodnoty momentů jsou sníženy o požadovanou hodnotu, viz. Realizace chodu pohonů.

### 6.2.1.3 Realizace chodu pohonů

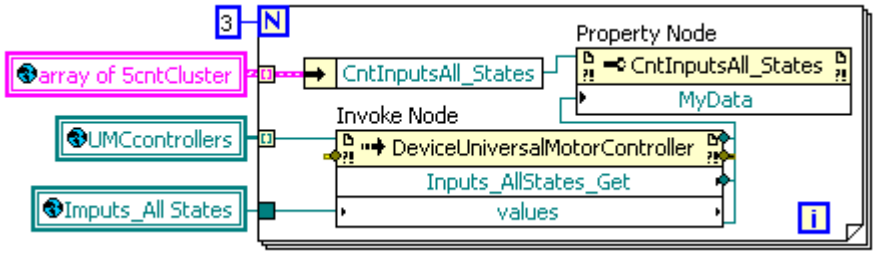
Realizace chodu pohonů představuje významnou část této diplomové práce, viz. blokové schéma příloha 11. Stěžejním úkolem této programové části je:

- Vytvoření zásobníku (FIFO),
- Odstartování chodu pohonů,
- Čtení měřených dat a ukládání do fronty,
- Monitoring kritických chyb.

Ve vývojovém schématu, viz. obr. 27 je realizace chodu pohonů reprezentována blokem C. Vytvořené blokové schéma obsahuje 19 *SubVI* umístěných ve smyčce *While Loop*, které za běhu experimentu paralelně pracuje s událostí B) Práce se zaznamenávanými daty obsažené v kapitole 6.2.1.2 a tento běh je synchronizován pomoci funkce *Rendezvous* a lokální proměnné „IsRuning“, viz. kap. 6.2.1.5. Smyčka *While Loop* obsahuje přepínač *Case Structure*, který má dva stavy čtení dat obsahující informace o pohonech do *.NET Containers* a uvedení pohonů do chodu, při kterém dochází ke sběru dat ze zařízení.

**A) Čtení měřených dat obsahující informace o pohonech**

Programový obsah tohoto *Case Structure* je aktivní v případě, kdy není odstartován chod pohonů a obsluhuje číselný obsah *.NET Containers* umístěných na předním panelu v záložce „Reference position settings“, viz. příloha 3 - A. Obsah tohoto *.NET Containers* zde byl umístěn z důvodu kontroly číselných údajů o pohonech (rychlosti, polohy, napětí, momentu, atd.) při nastavování výchozích poloh pohonů. Pomocí programových bloků obsažených ve *for cyklu* je využívána prostřední vrstva software obr. 28 a tím je docíleno čtení měřených dat o pohonech.




*Obr. 28 Cyklus for vyčítající číselné údaje o pohonech do .NET Containers.*

**B) Uvedení pohonů do chodu a sběr dat ze zařízení**

V této části *Case Structure* dojde k vytvoření zásobníku FIFO a zjištění logického stavu tlačítek „Read torque from sensor“. V *Case Structure* jsou obsaženy dvě stěžejní programové struktury:

- *SubVI* „Synchronization“,
- *Stacked Sequence* skládající se ze čtyř záhlaví.

**SubVI „Synchronization“**

Do *SubVI* „Synchronization“ vstupuje vytvořený zásobník FIFO a informace o stavu tlačítek „Read torque from sensor“. Toto *SubVI*, viz. příloha 12 je tvořeno pomocí postupně v sobě vnořených cyklů. Řazeno od nejvýše postaveného:

- *While Loop*,
- *Case Structure*,
- *Time Loop*.

Cyklus *While Loop* obsahuje globální proměnnou, ze které jsou čteny uživatelem nastavené parametry pro jednotlivé pohony (rychlosti a mezní úhly natočení misek). Tyto parametry vstupují do *SubVI* sloužící k odstartování chodu pohonů. *SubVI*, také umožňují díky využití naprogramovaných funkcí obsažených v prostřední vrstvě software kontrolovat pohyb pohonu. Jedním z výstupů tohoto *SubVI* je logická proměnná, která obsahuje stav:

- „*true*“ pokud došlo k pohybu pohonu do uživatelem nastavené polohy,
- „*false*“ pokud pohon nedosáhnul polohy požadované uživatelem.

Tento výstupní stav má vliv na následující *Case Structure*. Díky zvyšujícím se iteracím každého z provedených cyklů *While loop* je umožněno kontrolovat, jaký cyklus právě probíhá. Kontrola počtu proběhnutých cyklů je využita pro ukončení chodu pohonů a vyčítání dat ze zařízení v případě, je-li dosaženo požadovaného počtu zátěžných cyklů.

*Case Structure* je složen ze dvou stavů, které reagují na vzniklé události *SubVI*, které slouží pro odstartování chodu pohonů:

- „*false*“ – dojde k okamžitému zastavení dalšího možného chodu pohonů a uživateli je nahlášena chyba,
- „*true*“ – obsahuje poslední ze zmíněných cyklů *Time Loop*.

Cyklus *Time Loop* slouží k opakovanému provádění části programu v uživatelem zvolených časových periodách (nastavení rychlosti ukládání a vyčítáni dat v grafech). Obsahuje:

- Vytvořené *SubVI*, pomocí něhož je využito prostřední vrstvy software pro získání funkcí na výčet dat ze zařízeni:
  - Rychlost pohonů,
  - Úhel natočení misek,

    - Momentů.
- Informaci o stavu tlačítek, která je vstupem do *SubVI* „Synchronization“, jak bylo řečeno na začátku tohoto bodu. Pokud uživatel zvolí volbu získávání momentů pomocí senzoru, dojde k využití funkce *DAQmx Read* (obecný význam, viz. kap. 5.1) a na základě provedených kalibrací, viz. kap. 7 se na místo momentů získávaných z podřízené vrstvy vyčítají momenty za pomoci senzorů umístěných na hřídeli.
- Zjištění aktuálního času od počátku odstartování experimentu. Z důvodu stejného načasování odečítaných údajů a přesného časového údaje, kdy byly vzorky pořízeny, je využita funkce *Occurrences* (základní popis funkce, viz. kap. 5.2.2)

Všechny vyčtené parametry se uloží do vytvořeného zásobníku FIFO.

Smyčky chodu pohonů a zároveň výčet parametrů obsažený v *SubVI* „Synchronization“ se ukončí v případech, pokud dojde v cyklu:

- *Timed Loop* k:
    - Vzniku chyby při odečítání parametrů ze zařízení,
    - Změně stavu globální proměnné „zastavit“,
    - Změně stavu funkce *Rendezvous* určené k synchronizaci chodu software.
- *While Loop* k:
    - Vzniku jakékoli předchozí události v rámci cyklu *Timed Loop,*
    - Změně stavu globální proměnné „zastavit“,
    - Stavu, kdy požadovaný počet zátěžných cyklů je roven provedeným cyklům.

### Stacked Sequence

Pokud dojde k ukončení *SubVI* „Synchronization“, a to buďto z důvodu pozastavení experimentu nebo jeho ukončení uživatelem (tlačítko „Stop“, požadovány zátěžný cyklus je roven aktuálnímu cyklu), tak se následně vždy provede *Stacked Sequence* skládající se ze čtyř záhlaví:

- Zastavení chodu všech pohonů, nastavení lokální proměnné na stav „false“ tzn., zamezí se provádění smyček pro řízení pohonů a vyčítání dat ze zařízení,
- Dojde k uvedení tlačítek a editačních polí do výchozího stavu,
- Vyprázdnění možných dat, která se nevyčetla ze zásobníku FIFO a následné uložení těchto dat do *.tdms,
- Pokud nastal stav po pauze, tak tento čtvrtý rámec neobsahuje žádné programové bloky. V případě stavu, kdy experiment byl zastaven (ne po pauze) uživatelem, pak se v tomto rámci provede uložení časového záznamu, kdy byl experiment ukončen a následně dojde k uzavření souboru *.tdms a zásobníku FIFO.

#### 6.2.1.4 Ukončování

Pokud uživatel uzavírá vytvořený software dochází k:

- Ukončení zásobníku FIFO,
- Ukončení synchronizační funkce *Rendezvous*,
- Ukončení *DAQmx*,
- Smazání dat obsažených v online grafech,
- Uvedení ovládacích a editačních prvků na předním panelu do výchozího stavu.

#### 6.2.1.5 Shrnutí realizace paralelního běhu řídicího software

Jednotlivé části programu popsané v kapitolách 6.2.1.2, 6.2.1.3 a 6.2.1.4, jsou navzájem synchronizovány za pomoci funkce *Rendezvous* (obecné vlastnosti této funkce, viz. kap. 5.2.1) a využitím lokální proměnné „IsRuning“ zmíněné v předešlém textu. Hlavní snahou při synchronizaci bylo vytvořit logické propojení mezi kódem programu popsaným kapitolou 6.2.1.2 zabývající se reakcí na vzniklé události a smyčkou sloužící k realizaci chodu pohonů, viz. kap. 6.2.1.3. Popis základní myšlenky návrhu synchronizace:

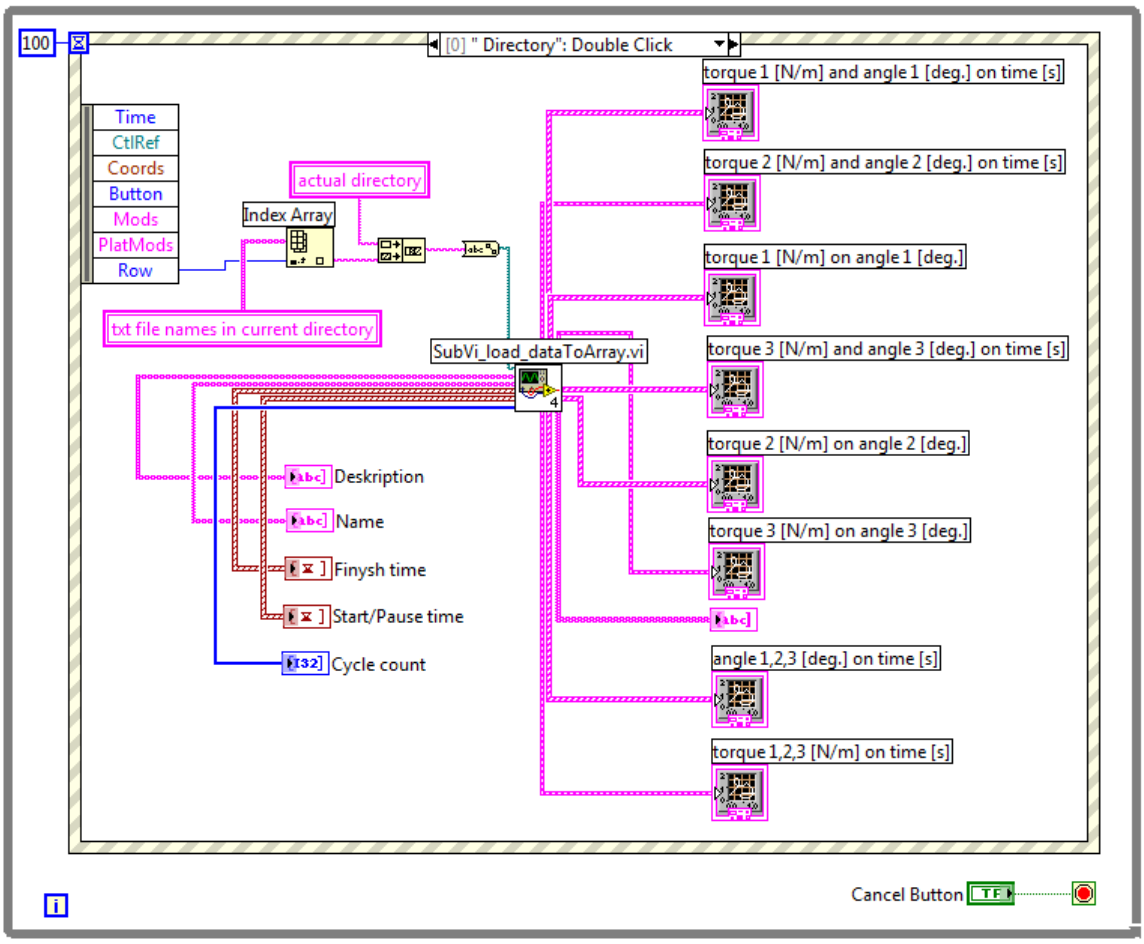
- Při vzniku události *Button Start* se za pomoci změny logického stavu lokální proměnné „IsRuning“ uvede do chodu smyčka pro realizaci chodu pohonu. V této smyčce dochází k vyčítání dat ze zařízeni a ukládání do zásobníku FIFO. Zároveň je odstartována smyčka obsažená v kapitole pro reakce na vzniklé události. V této smyčce *TimeOut* jsou data čtena ze zásobníku a ukládána do datového souboru formátu *.tdms. Data jsou dále reprezentována uživateli jako číselné hodnoty na předním panelu a v grafech.
- Pokud dojde k pozastavením, je v události „Pauze“ změněn stav lokální proměnné „IsRuning“ na stav, při kterém se zastaví smyčka pro realizaci chodu pohonů a zároveň v události *TimeOut* se přestane provádět čtení dat ze zásobníku FIFO. Doposud čtená data se přestanou ukládat do formátu *.tdms. Tento stav umožní uživateli změnit nastavení experimentu. Uživatel po pauze odstartuje experiment a je opakován předchozí bod.
- Jakmile uživatel ukončí běh experimentu (vznikem události *Stop*), dojde za pomoci globální proměnné „Zastavit“ k ukončení *SubVI* „Synchronization“. Výsledkem je zjištění stavu funkce *Rendezvous*, pomocí kterého se ukončí celá smyčka realizací chodu pohonů. V tomto bodu uživatel může vytvořit a spustit nový experiment (první bod), nebo uživatel může zvolit možnost program uzavřít.
- Při uzavíráni programu je ukončen cyklus, ve kterém je řešeno odchytávání vzniklých událostí a pomocí funkce *Rendezvous* a je proveden obsah popsaný, viz. kapitola 6.2.1.4.

### 6.2.2 Blokové schéma software pro offline zobrazování dat

Chod celého programu pro zobrazování dat uložených ve formátu *.tdms je založen na VI *Event Structure* obr. 29, který reaguje na:

- Událost *Value Change* pro načtení datového úložiště do VI *Listbox*,
- Událost *Mouse Down* k otevření *TDMS* prohlížeče (*TDMS Viewer*),
- Událost *Double Click* pro zobrazení uložených dat ve formátu *.tdms z VI *Listbox*

Pro zobrazení uložených dat (událost *Double Click* v VI *Listbox*) je umístěno *SubVi*, které pracuje s vybraným souborem. *SubVi*, viz. příloha 13 dokáže pomocí funkcí určených pro práci s poli a I/O metod data rozdělit tak, že můžeme pracovat pouze s hodnotami jedné zaznamenané veličiny. Takto oddělená data lze snadno kombinovat pro získání požadovaných závislostí, které jsou následně zobrazeny v grafech, tabulce a informačních polích, viz. kap. 6.1.2.




*Obr. 29 Event Structure obsluhující výskyt čtyř událostí.*

Prostředí NI LabVIEW 8.6 nabízí nástroj pro prohlížení zaznamenaných dat ve formátu *.tdms. Pomocí tlačítka „TDMS Viewer“ umístěného na předním panelu reagujícího na událost *Mouse Down* dojde otevření *TDMS*, viz. kap. 6.1.2.

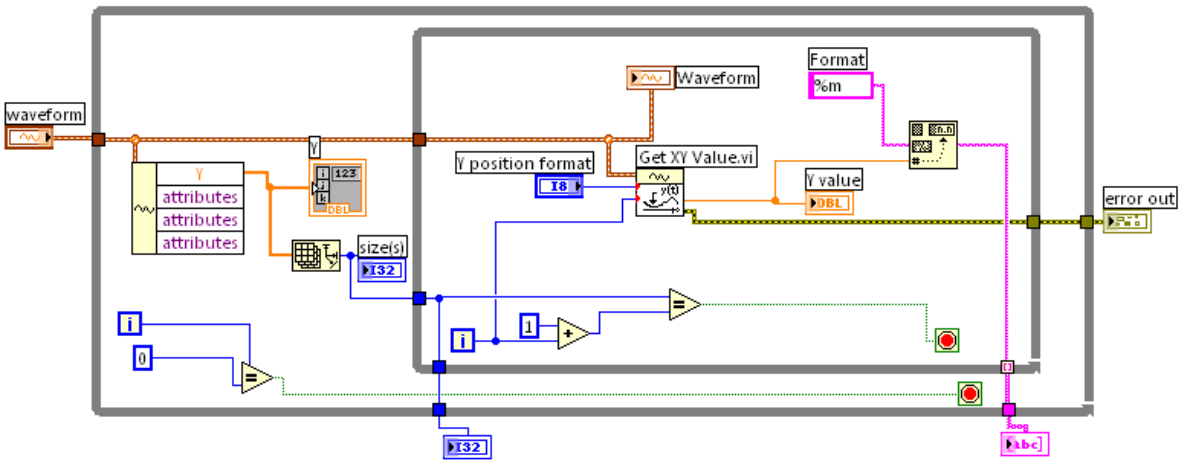
### 6.2.3 Blokové schéma software pro konverzi dat

Blokové schéma obsahuje cyklus *While Loop*, který se při spuštění software neustále opakuje s intervalem pětiset milisekund. V tomto cyklu je obsažena smyčka *Event Structure* čekající na výskyt události *Value Change* vyvolaná stiskem tlačítka „Konvert“ umístěným na předním panelu tohoto software. Na počátku události *Value Change* je přistupováno k uloženým datům pomocí funkcí *TDMS*. Dle požadavků uživatele, viz. kap. 4.3 se data převáděná z *.tdms do univerzálního formátu *.txt rozdělují na:

- Data obsahující nastavení experimentu,
- Data měřená ze zařízení.

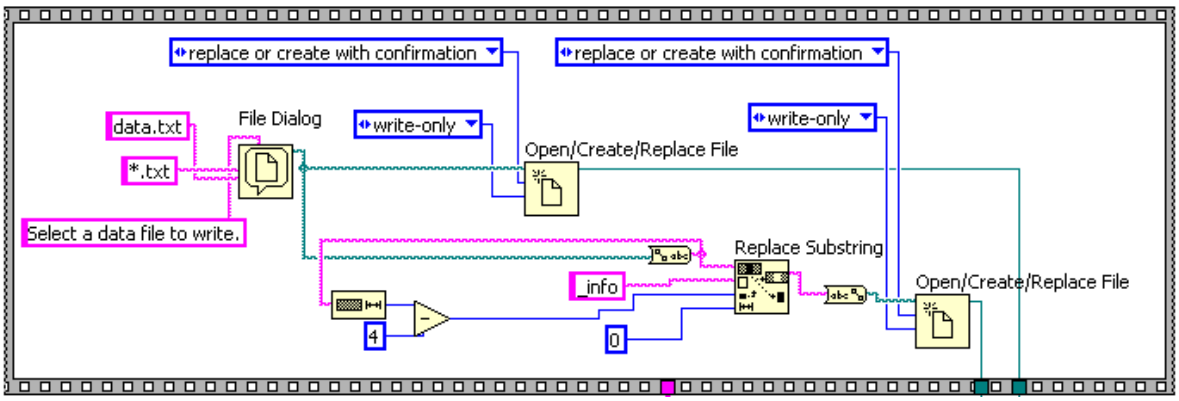
Data naměřená ze zařízení je nutno separovat pomocí vytvořeného *SubVI*, viz. obr. 30 tak, aby vytvořila jedenáct sloupců. V každém sloupci musí být obsažena data udávající hodnoty jen jedné z jedenácti veličin, podrobný popis, viz. kap. 6.1.3. Dále jsou vytvořeny dvě *SubVI*. Do prvního vstupují separovaná data, která jsou následně ukládána do formátu *.txt. Vstupy druhého *SubVI* jsou data načtená pomocí funkcí *TDMS* (obsahující nastavení

experimentu). Toto *SubVI* obsahuje rozsáhlý mechanizmus, který separuje data a zároveň je převádí do formátu *.txt.




*Obr. 30 Část programového kódu pro oddělení požadovaných dat.[3]*

Závěrem je uživateli nahlášena zpráva o uložení dat a to do dvou souborů *.txt, viz. kap. 6.1.3. Programový mechanizmus, pomocí něhož jsou alokovány oba zmíněné soubory, je znázorněn na obr. 31.




*Obr. 31 Vytvoření souborů pro následné uložení převedených dat.*

# 7 KALIBRACE

Vytvořený software umožňuje dva způsoby zjištění momentů vznikajících na testovaných páteřních prvcích a to pomocí:

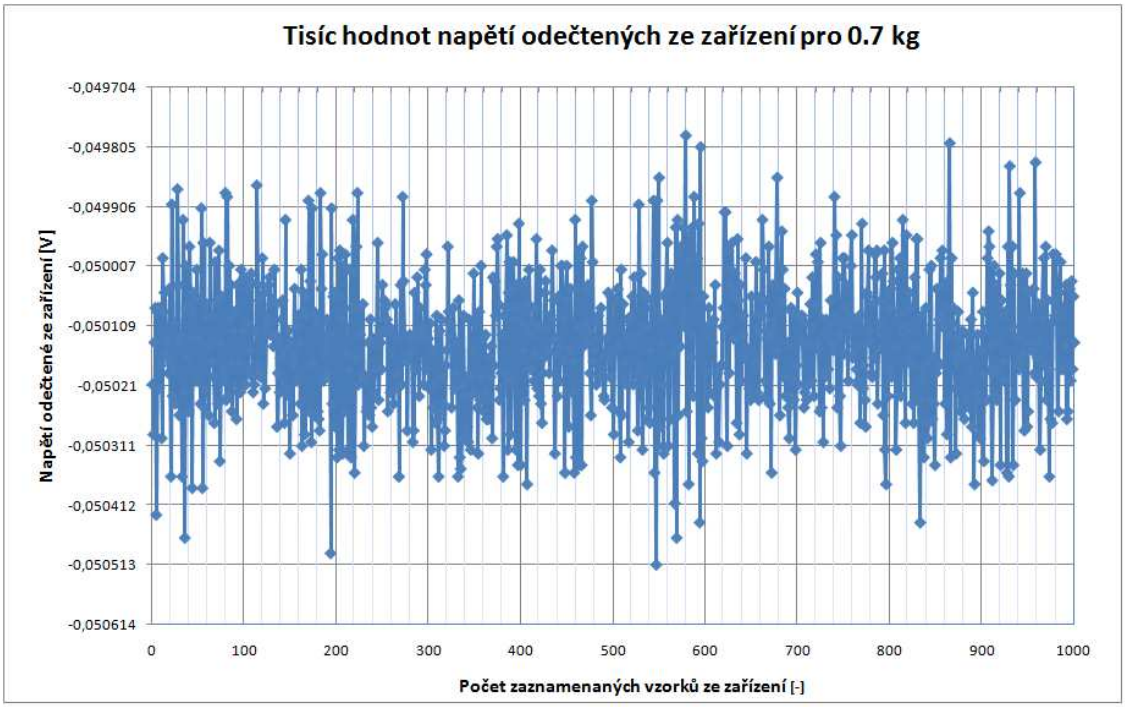
- Přepočtu z velikosti napětí odebíraného z pohonů (problematika řešena v rámci podřízené vrstvy software, není předmětem této práce).
- Tenzometrů umístěných na hřídeli pohonu, které měří velikost deformace (měřenou fyzikální veličinou je odpor), pomocí níž je možné určit velikost mechanického napětí, elektrického napětí a konečně i působícího momentu.

V rámci této diplomové práce byl nad rámec zadání řešen především druhý způsob získávání momentů. Kapitola 2.3 obsahuje výpis zvolených prostředků sloužících k zajištění uživatelem požadovaného momentu prostřednictvím senzorů umístěných na hřídeli mezi pohony a miskou v ose flexe. Z důvodu dosažení dostatečně přesných hodnot působících momentů bylo nutné přistoupit ke kalibraci. Tím je myšleno zpřesnění přepočtu naměřených hodnot na hodnoty působících momentů.

## 7.1 Použitý přístup

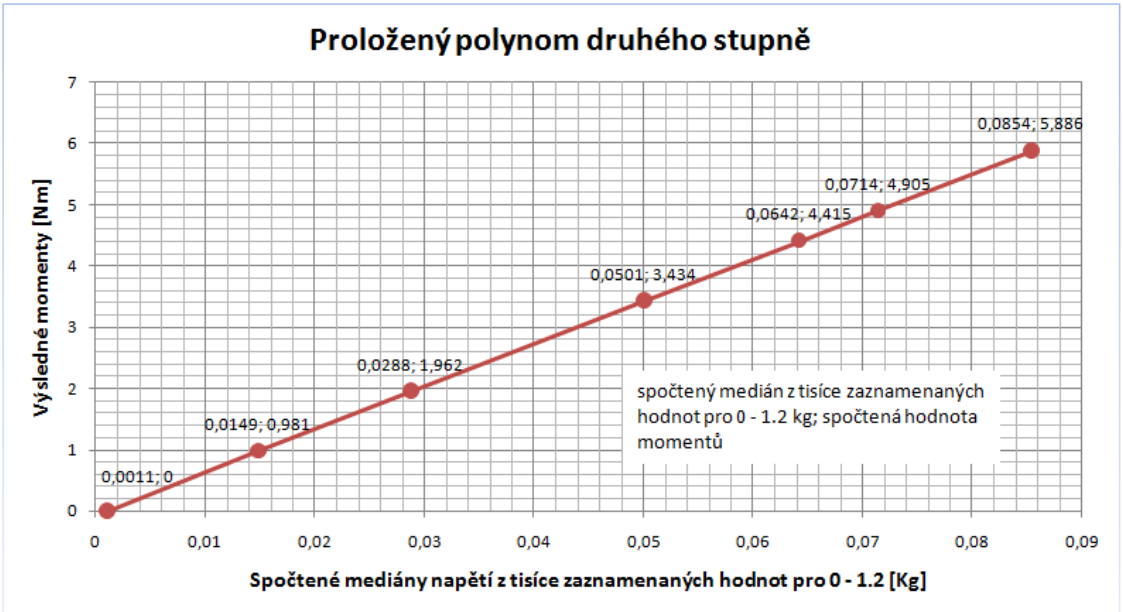
Předmětem zájmu byly pouze momenty vyvozované v ose flexe. Pro zvolený pohon byl na misku z jedné strany upevněn uzavřený čtvercový ocelový profil o délce 0.5 m. Na jeho konci bylo postupně zavěšováno závaží o zvolené hmotnosti (0 kg, 0.2 kg, 0.4 kg, 0.7 kg, 0.9 kg, 1 kg, 1.2 kg). Volba délky profilu a maximální volené hmotnosti závaží vychází z konstrukčního omezení pohonů a převodovek obsažených na zařízení. Takto sestavené konstrukční řešení umožnilo zatížit pohon známými hodnotami momentů a k nim za pomocí použitého zařízení, viz. kap. 2.3 zjistit ze senzorů příslušné hodnoty elektrického napětí. Měření bylo prováděno za stavu, kdy je spuštěn řídicí software, ale není odstartován experiment, tj. misky nejsou v pohybu.

Z důvodu eliminace nepřesností vzniklých při měření napětí, bylo pro každý známý moment přečteno celkem 1000 hodnot. Pro tyto účely bylo do řídicího software doprogramováno řešení, které po zapnutí software automaticky uloží požadovaný počet hodnot napětí. Průběh zaznamenaných napětí pro konkrétní zatížení (0.7 kg) je patrný z obr. 32.




*Obr. 32 Ukázka tisíce zaznamenaných hodnot napětí při zatížení 0,7 kg závaží na rameni 0,5 m.*

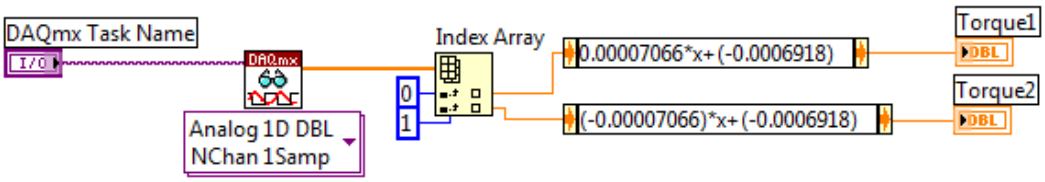
Pro každou zvolenou hodnotu závaží bylo snímané napětí uloženo do samostatného textového souboru a z těchto hodnot určen medián. Celkem 6 mediánů bylo následně proloženo pomocí programu Octave polynomem druhého stupně, viz. obr. 33. Řád polynomu byl zvolen na základě konzultace se školitelem této práce.




*Obr. 33 Proložení spočtených momentů (a jim odpovídajících změřených napětí) polynomem druhého stupě.*

Vypočtené koeficienty zmíněného polynomu jsou v předloženém software použity pro přepočet ze zjištěného napětí na působící moment. Komunikace mezi

použitými prostředky a řídicím software vytvořeným v prostředí NI LabVIEW 8.6 je umožněna pomocí funkcí *DAQmx*, viz. kap. 5.1. Na obr. 34 je znázorněna funkce *DAQmx*, pomocí které jsou zjištěny hodnoty napětí ze zařízení a následně násobeny spočteným polynomem druhého řádu, tak aby byl získán konkrétní aktuálně působící moment.




*Obr. 34 Blokové schéma funkce DAQmx pro přečtení hodnoty ze senzorů a následný přepočet.*

# 8 ŘEŠENÍ PROBLÉMŮ VZNIKLÝCH PŘI REALIZACI

Při realizaci software vytvořeného v rámci této diplomové práce vznikla celá řada problémů. Některé ze vzniklých problémů se ukázaly až při provádění dlouhodobého testování. V této kapitole jsou popsány stěžejní problémy, které bylo nutné při realizaci řídicího software řešit. Jejich řešení lze považovat za přínos této práce.

## 8.1 Hierarchie ukládání

Na základě požadavků pracovníků ÚMTMB, viz. kap. 4, bylo vypracováno řešení problému s hierarchií ukládaní dat. Data, se kterými se v rámci řídicího software pracuje, můžeme rozdělit do tří skupin. Hlavní skupinou jsou data, která jsou odečítána za chodu experimentu. Další dvě skupiny dat slouží k uchování nastavení pohonů a obsahují informace o experimentu. Tyto tři skupiny musí být mezi sebou logicky propojeny a to tak, že při vzniku souboru *.tdms dojde k uložení dat obsažených v *.NET Containers* a to pouze v případě, kdy došlo k prvnímu odstartování v rámci jednoho experimentu. Data získávaná za běhu experimentu jsou ukládána po dobu běhu. Pokud dojde k pauze a následnému spuštění experimentu, na předchozí data je navázáno. Data týkající se popisu experimentu zadané od uživatele jsou uložena automaticky, při spuštění experimentu a při pozastaveni nebo změně těchto údajů za chodu experimentu. Takto ukládaná data bylo nutné chránit proti výpadku el. proudu, přerušení experimentu či chybě na úrovni operačního systému.

Takový logický sled zaznamenávání dat při průběhu dlouhodobých experimentů bylo nutno řešit pomocí správné posloupnosti událostí a zvolených programových nástrojů např. globální, lokální proměnná. Výsledkem je komplexní mechanizmus, který uživateli poskytuje uložení veškerých zmíněných skupin dat tak, aby byla zaručena jejich maximální bezpečnost za použití technologii (*TDMS*, zásobníku FIFO), které nabízí programové prostředí NI LabVIEW 8.6. Takto vytvořené řešení je popsáno v kap. 6.2.1.

## 8.2 Synchronizace smyček pohybu motorů a ukládání dat

Problém se synchronizací smyček byl jedním ze stěžejních úkolů řešených v této diplomové práci. Z důvodu ověření nových technologii, které nabízí programové prostředí NI LabVIEW 8.6 byla vytvořena samostatná aplikace. Tato aplikace měla za úkol ověřit a simulovat vhodnost použití funkcí navrhnutého řešení. Řešením se ukázalo využít funkce *Rendezvous* a lokální proměnné, viz. kap. 6.2.1.5. Za použití těchto nástrojů došlo k synchronnímu pohybu dvou stěžejních smyček programu.

## 8.3 Ukládání velkého objemu dat

Výstupem vytvořeného řídicího software jsou zaznamenaná data pocházející ze zařízení na dlouhodobé experimenty. Jelikož se jedná o velké množství dat, které je nutné ze zařízení ukládat, bylo nutné použití vyspělých nástrojů. Jedním z testovaných řešení se ukázalo jako nejspolehlivější využití technologie, kterou nabízí samo prostředí NI LabVIEW 8.6. Jedná se o funkci *TDMS*, která poskytuje velkou rychlost ukládání dat. Tento binární formát ukládání dat je významný i díky svojí široké škále možností nastavení,

viz. kap. 5.3. Díky vlastnostem poskytující *TDMS* byl vytvořen robustní mechanizmus, který zabezpečí ukládaná data.

Negativním poznatkem získaným při práci s formátem *TDMS* byla zkušenost, kdy pro dělení do logicky řazených kategorii byla použita řada stejných prvků, které se odlišovaly pouze hlavičkou. Pro potřebu načtení dat ve formátu *TDMS* bylo nutno k uloženým datům přistupovat postupně, tak jak byla uložena. Tímto vznikala posloupnost stejných funkcí a to s nutností informace v jakém pořadí došlo k ukládání těchto dat. Takto logicky řazená a uložená data jsou ve formátu *.tdms. Pro potřeby uživatele je umožněn převod pomocí vytvořeného software do formátu *.txt.

## 8.4 Problémy vzniklé při dlouhodobých testech vytvořeného software

Jedním z předpokladů a požadavků ze strany pracovníků ÚMTMB je, aby vytvořený řídicí software byl schopen dlouhodobých experimentů. Uživatelem byla stanovena dvou hodinová doba, po kterou musí být zařízení schopno bezproblémově pracovat. Toto časové kritérium vyplynulo z faktu, že zařízení je určeno pro zkoumání otěru páteřních segmentů. Jelikož se jedná o prvek, který rychle podléhá prostředí, tzn. vlivem působícího okolí dochází k osychání. Důsledkem osychání je nežádoucí změna mechanických vlastností testovaného páteřního segmentu. V rámci ověření vytvořeného software byla tato dvouhodinová doba předimenzována na osm hodin. Čtyřnásobným předimenzováním je uživateli zaručen jeho požadavek.

Po vytvoření a průběžném krátkodobém testování funkčnosti řídicího software bylo přistoupeno k ověření vytvořeného software na dlouhodobých testech. Provedeny byly čtyři dlouhodobé testy trvající osm hodin. V každém z nich se ukázalo, že dochází k nárůstu spotřeby paměti a využití výpočetního výkonu procesoru PC. Každou hodinu běžícího experimentu byl nárůst využití paměti průměrně navýšen o 60 MB a využití CPU o 12%.

### 8.4.1 Návrh a oprava software na základě výsledků dlouhodobých testů

Z důvodu, které jsou rozepsány v předešlé kapitole, byla nutná optimalizace vzniklého návrhu blokového schématu. Ukázalo se, že nedocházelo k uvolňování vytvářených instancí a díky velkému počtu opakování docházelo k nárůstu spotřeby uvedených počítačových zdrojů. Díky opravě, která spočívala v uvolnění vznikajících instancí, byl tento problém odstraněn. Při vývoji software bylo zdokonalováno používání technologii, které nabízelo prostředí NI LabVIEW 8.6. Závěrem práce bylo vytvořeno sjednocení používání všech technologii a tím došlo ke zvýšení robustnosti vytvořeného blokového schématu. Došlo k nahrazení jiných řešení, kdy nebylo zapotřebí aplikování tak velkého využívání lokálních proměnných.

# 9 ZÁVĚR

Práce je členěna do dvou logicky řazených částí. První část je pojata jako teoretická a obsahuje pět kapitol. První kapitola představuje úvod do celé problematiky, která byla řešena v rámci této závěrečné práce. Ve druhé kapitole je popsáno zařízení určené k dlouhodobým testům na páteřních prvcích. Následující kapitola podrobněji popisuje použitý software. Kladené požadavky na vytvořený software, které vznikly ze strany pracovníků ÚMTMB jsou uvedeny v kapitole čtyři. Poslední kapitola obsažena v této teoretické části popisuje stěžejní použité programové nástroje obsažené v NI LabVIEW 8.6. Mezi tyto nástroje bezesporu patří funkce pro synchronizaci toku dat vláken, práce s daty pomocí *TDMS*, nástroje pro použití .NET kódu v NI LabVIEW 8.6, atd. Druhá část této diplomové práce zahrnuje veškerou vlastní práci autora, viz. kap. 6, 7 a 8. Kapitola 6 se zabývá podrobným popisem vytvořeného software. Jedná se jak o popis vytvořeného uživatelského rozhraní, viz kap. 6.1, tak o popis blokového schématu tvořícího vlastní programový kód vytvořeného software, viz. kap. 6.2. Kapitola 7 se zabývá výpočtem momentů vznikajících v osách flexe a vhodným způsobem přepočtu z měřeného napětí. Poslední kapitola shrnuje řešení stěžejních problémů vzniklých při tvorbě software.

Výsledkem této práce je funkční software splňující veškeré požadavky kladené ze strany pracovníků ÚMTMB. Takto vytvořený software na základě požadavků umožňuje:

- Plánovat experimenty,
- Spouštět, pozastavovat experimenty,
- Ukládat data získávaná za běhu experimentu do formátu *TDMS*,
- On-line zobrazení zaznamenávaných dat,
- Zasahovat do již běžícího experimentu,
- Zobrazovat informace o případných problémech,
- Vytvářet popis experimentu, který se ukládá s daty,
- Ukládat veškeré informace o nastavení parametrů experimentu potřebné pro případné opakování experimentu,
- Nastavení výchozích poloh pohonů pro odstartování chodu experimentu,
- Ochranu stroje proti možným kolizním nastavením pohybu pohonů.

Kromě hlavního řídicího software jsou do předloženého řešení začleněny i dvě pomocné aplikace, se kterými uživatel může za chodu experimentu pracovat. První umožňuje offline zobrazování uložených dat, viz. kap. 6.1.2. Cílem software je offline analýza uložených dat ve formátu *TDMS* za účelem dalšího výzkumu a porovnávaní. Data jsou zde zobrazována v tabulce a v grafech, které umožňují jejich podrobnou analýzu. Druhá slouží pro převod uložených dat do univerzálního formátu, viz. kap. 6.1.3. Umožňuje převod uložených dat z formátu *.tdms do formátu *.txt a to převážně z důvodu zpracování těchto dat pomocí jiných programů např. Matlab.

Nad rámec zadání bylo umožněno získávání momentů vznikajících na miskách přístroje v ose flexe a to pomocí tenzometrů měřících velikost deformace na hřídeli mezi miskami a pohony. K tomuto účelu byl použit hardware popsaný v kapitole 2.3. V rámci práce došlo také ke zpřesnění přepočtu v rámci použitého řešení, viz. kap. 7.1. Tímto způsobem je uživateli poskytnuta druhá možnost získávání aktuálně působících momentů. Oproti momentům získaných výpočtem za pomoci podřízené vrstvy software poskytuje tato možnost výsledky s větší přesností (lépe odpovídající realitě).

Při realizaci software došlo k několika problémům. V kapitole 8 jsou popsány ty, jejichž řešení mělo velký vliv na další postup tvorby práce. Prvním z uvedených problémů bylo zajištění robustního způsobu ukládání dat dle stanovených požadavků, a to i v případě výpadku software, přerušení experimentu či chybě na úrovni operačního systému. Druhým

stěžejním problémem v rámci práce byla synchronizace smyček v nichž je realizován pohyb motorů a ukládání dat za pomoci funkcí nabízející prostředí NI LabVIEW 8.6, viz. kap. 8.2. Jelikož konečným výstupem vytvořeného software jsou data snímaná ze zařízení určeného k dlouhodobým testům, vznikla rovněž otázka ukládání velkého objemu dat. Jako řešení se ukázalo využití funkcí pro práci s datovým formátem *TDMS* implementované v NI LabVIEW 8.6. Při použití zmíněného formátu pro velké množství dat, které jsou ve velké míře členěna do kategorií, je využití funkcí pro manipulaci s *TDMS* poněkud těžkopádným avšak z hlediska poskytovaných výhod nevyhnutelným řešením, viz. kap. 8.3. K řešení všech zmíněných problémů bylo úspěšně použito nástrojů, které jsou popsány v teoretické části, viz. kap. 5.

Takto vytvořený software byl podroben několika dlouhodobým osmi hodinovým testům, viz. kap. 8.4. Na základě jejich výsledků byl zjištěn nárůst spotřeby paměti a využití výpočetního výkonu procesoru PC s postupem času experimentu. Tento problém se podařilo vyřešit částečnou úpravou původního návrhu software, viz. kap. 8.4.1.

Výsledkem této diplomové práce je zcela funkční software použitelný pro plánování a zaznamenávání dlouhodobých experimentů na přístroji sloužícím pro zkoumání vlivu zatížení na velikosti otěru páteřních segmentů. Předmětem této diplomové práce byla pouze nejvyšší vrstva z celkem tří vrstev předkládaného software. Předložené řešení splňuje veškeré kladené požadavky. Před zahájením provozu podstoupilo vytvořené řešení celou řadu dlouhodobých testů, které prokázaly jeho funkčnost. V současné době je testovací zařízení k dispozici v laboratoři mechaniky těles ÚMTMB FSI VUT, kde je používán k experimentům a zaznamenána data slouží pro výzkum problematiky v dané oblasti.

Do budoucna je možnost další spolupráce ústavů ÚAI a ÚMTMB, a to z důvodu vývoje dalších přístrojů určených pro zkoumání otěru prvků např. přístroj pro zkoumání vlivu otěru na kloubních segmentech.

## Poděkování

Publikovaných výsledků bylo dosaženo za podpory ministerstva školství, mládeže a tělovýchovy České republiky, výzkumný záměr MSM 0021630518 "Simulační modelování mechatronických soustav".

# SEZNAM POUŽITÝCH OBRÁZKŮ

Obr. 1 Konstrukčně zdokonalené zařízení pro zkoušení páteřních segmentů ..... 13
Obr. 2 Celkový pohled na inovovanou sestavu zařízení pro zkoušení páteřních segmentů 13
Obr. 3 Detail uchycení motoru pohonu torze ..... 14
Obr. 4 Detail uchycení motoru pro flexi ..... 14
Obr. 5 Terminál NI-SCC-68 ..... 15
Obr. 6 Karta NI-PCI6220 ..... 15
Obr. 7 Měřící sestava využívající DAQ kartu ..... 23
Obr. 8 Paleta funkcí pro práci s DAQmx. ..... 24
Obr. 9 Funkce DAQmx Start Task ..... 25
Obr. 10 Použití funkce DAQmx Start Task ..... 25
Obr. 11 Funkce DAQmx Stop Task. ..... 26
Obr. 12 Funkce DAQmx Read ..... 26
Obr. 13 Čtyři možnosti použití DAQmx Read ..... 26
Obr. 14 Paleta funkcí pro práci s rendezvous ..... 27
Obr. 15 Zavedeni Rendezvous ..... 27
Obr. 16 Náhled do SubVI User1 ..... 28
Obr. 17 Paleta funkcí pro práci s Occurrences. ..... 28
Obr. 18 Hierarchie TDMS ..... 29
Obr. 19 Práce s Queue – přidávání do zásobníku ..... 30
Obr. 20 Práce s Queue – vyjmutí ze zásobníku ..... 30
Obr. 21 Možnost použití lokální proměnné typu double ..... 31
Obr. 22 Možnost použití lokální proměnné typu boolean ..... 31
Obr. 23 Paleta funkcí .NET & ActiveX. ..... 32
Obr. 24 .NET Containers pro nastavení výchozích poloh motoru ..... 35
Obr. 25 Interface software pro konverzi dat. ..... 36
Obr. 26 Naměřená data, která jsou převedena do formátu *.txt ..... 37
Obr. 27 Schéma základního běhu software pro dlouhodobé testování. ..... 39
Obr. 28 Cyklus for vyčítající číselné údaje o pohonech do .NET Containers ..... 46
Obr. 29 Event Structure obsluhující výskyt čtyř událostí ..... 49
Obr. 30 Část programového kódu pro oddělení požadovaných dat ..... 50
Obr. 31 Vytvoření souborů pro následné uložení převedených dat ..... 50
Obr. 32 Graf závislosti všech zaznamenaných napětí na množství zaznamenaných dat .... 52
Obr. 33 Graf závislosti spočtených mediánu ze zaznamenaných dat k tomu příslušného momentu. ..... 52
Obr. 34 Blokové schéma funkce DAQmx pro zjištění momentu ze senzoru. ..... 53

# SEZNAM POUŽITÉ LITERATURY


[1] BŘEZINA, T.; a kol., *Simulation Modelling Of Mechatronics Systems I*, Faculty of Mechanical Engineering, Brno University of Technology. Brno. 2005, ISBN: 80-214-3344-X.

[2] BŘEZINA, T.; a kol., *Simulation Modelling Of Mechatronics Systems II*, Faculty of Mechanical Engineering, Brno University of Technology. Brno. 2006, ISBN: 80-214-3341-8.

[3] HEJČ, T.; *Uživatelské rozhraní pro řízení experimentů na biomechanických přístrojích*. Brno, 2007. 32 s., 8 s. příloh. Bakalářská práce na fakultě Strojního inženýrství VUT na Ústavu aplikované informatiky a řízení. Vedoucí bakalářské práce Ing. Vít Ondroušek.

[4] FLORIÁN, Z.; KOTEK, V.; VLK, M.; *Problémy experimentálního modelování na páteřních prvcích.* [pdf. dokument]. 2001 [cit. 7. 5. 2009], Dostupný z: http://www.umt.fme.vutbr.cz/osem/pdf/ean2001/Florian.pdf

[5] Technická podpora firmy NATIONAL INSTRUMENTS CORPORATION, citováno 14.3.2009, dostupný z: www.ni.com

[6] ŠTĚTINA, J.; JAROŠ, M.; RAMÍK, P.; *Virtuální laboratoř – Experimentální metody*. [pdf. dokument]. VUT FSI Brno. 2003 [cit. 6. 5. 2009]. Dostupný z: http://autnt.fme.vutbr.cz/lab/FAQ/labview/Skripta_Stetina2003.pdf

[7] ŽÍDEK, J.; *Grafické programování ve vývojovém prostředí LabVIEW*. VŠB-TU Ostrava. Říjen 2002

[8] HUSÁKOVÁ, A.; *Měřicí program LabVIEW*. [pdf. dokument]. [cit. 7. 2. 2009], Dostupný z: http://www.vscht.cz/ufmt/cs/lide/husakova.html

[9] TRAVIS, J.; KRING, J.; *LabVIEW for Everyone*, Graphical Programing Made Easy and Fun, Third Edition, Prentice Hall, July 27, 2006. 1032 p. ISBN 978-0-13-185672-1.

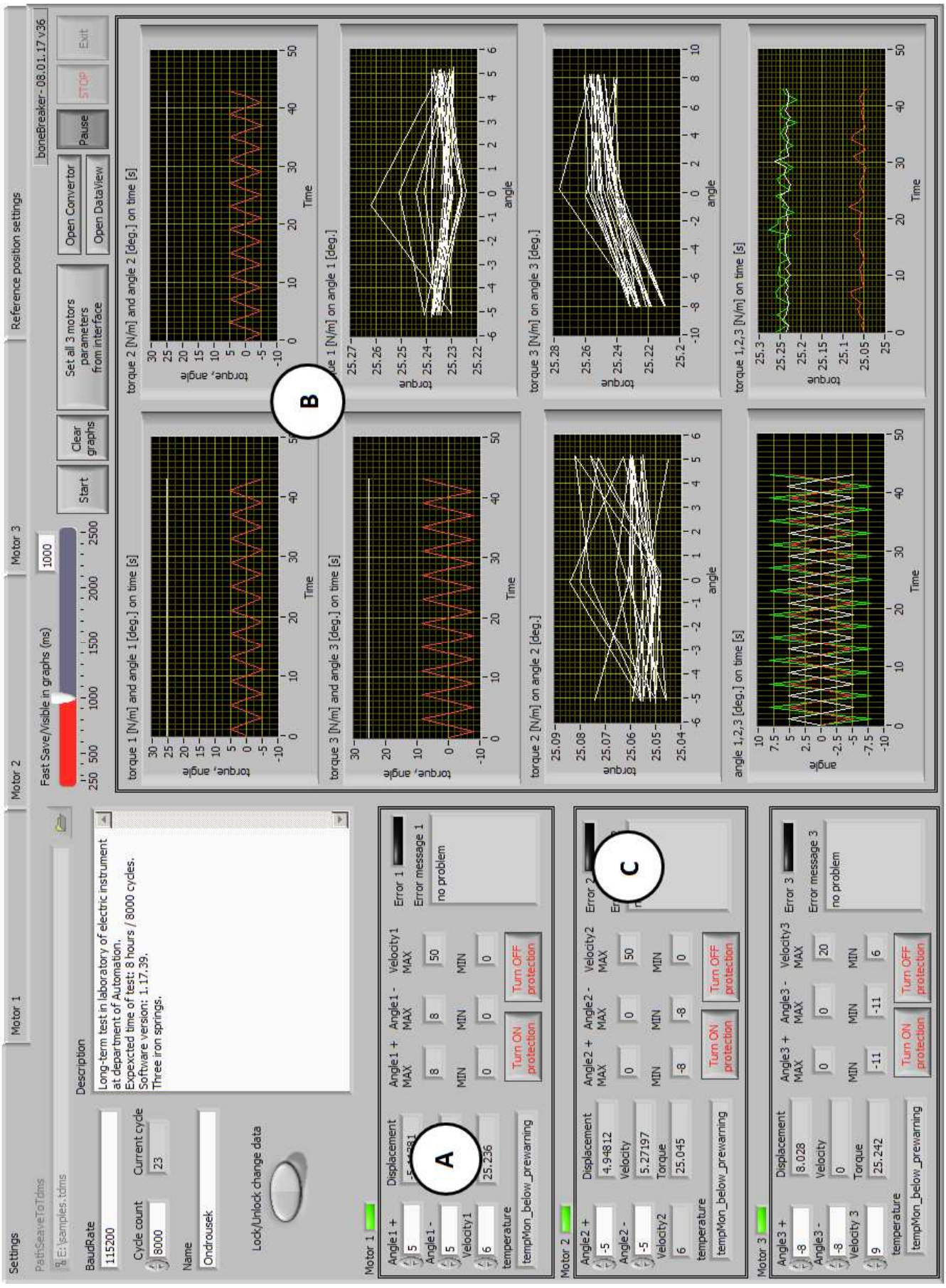
[10] Elektronická nápověda NATIONAL INSTRUMENTS LABVIEW 8.6.

# SEZNAM PŘÍLOH

Příloha č. 1 - Uživatelské rozhraní řídicího software záložka „settings“
Příloha č. 2 - Uživatelské rozhraní řídicího software záložka „motor“
Příloha č. 3 - Uživatelské rozhraní řídicího software záložka „Reference position settings“
Příloha č. 4 - Uživatelské rozhraní software pro offline zobrazováni dat záložka „Measured data“
Příloha č. 5 - Uživatelské rozhraní software pro offline zobrazováni dat záložka „Graphs moments and angles on times“
Příloha č. 6 - Struktura převedených dat do formátu *.txt
Příloha č. 7 - Blokové schéma inicializační části řídicího software
Příloha č. 8 - Blokové schéma programu prováděné při vzniku události „button start: mouse down“
Příloha č. 9 - Blokové schéma vzniklé události „timeout“
Příloha č. 10 - Část blokového schématu pro zapsání základních parametrů nastavení souvisejících s pohony do *.NET Containers*
Příloha č. 11 - Blokové schéma části programu zajišťující realizaci chodu pohonů
Příloha č. 12 - Blokové schéma *SubVI* Synchronization obsaženého v části realizace chodu pohonů
Příloha č. 13 - Blokové schéma pro načtení uložených dat ve formátu *TDMS* a jejich následnému rozdělení (do grafů, tabulek a informačních polí)
Příloha č. 14 - Přiložené miniDVD s obsahem:

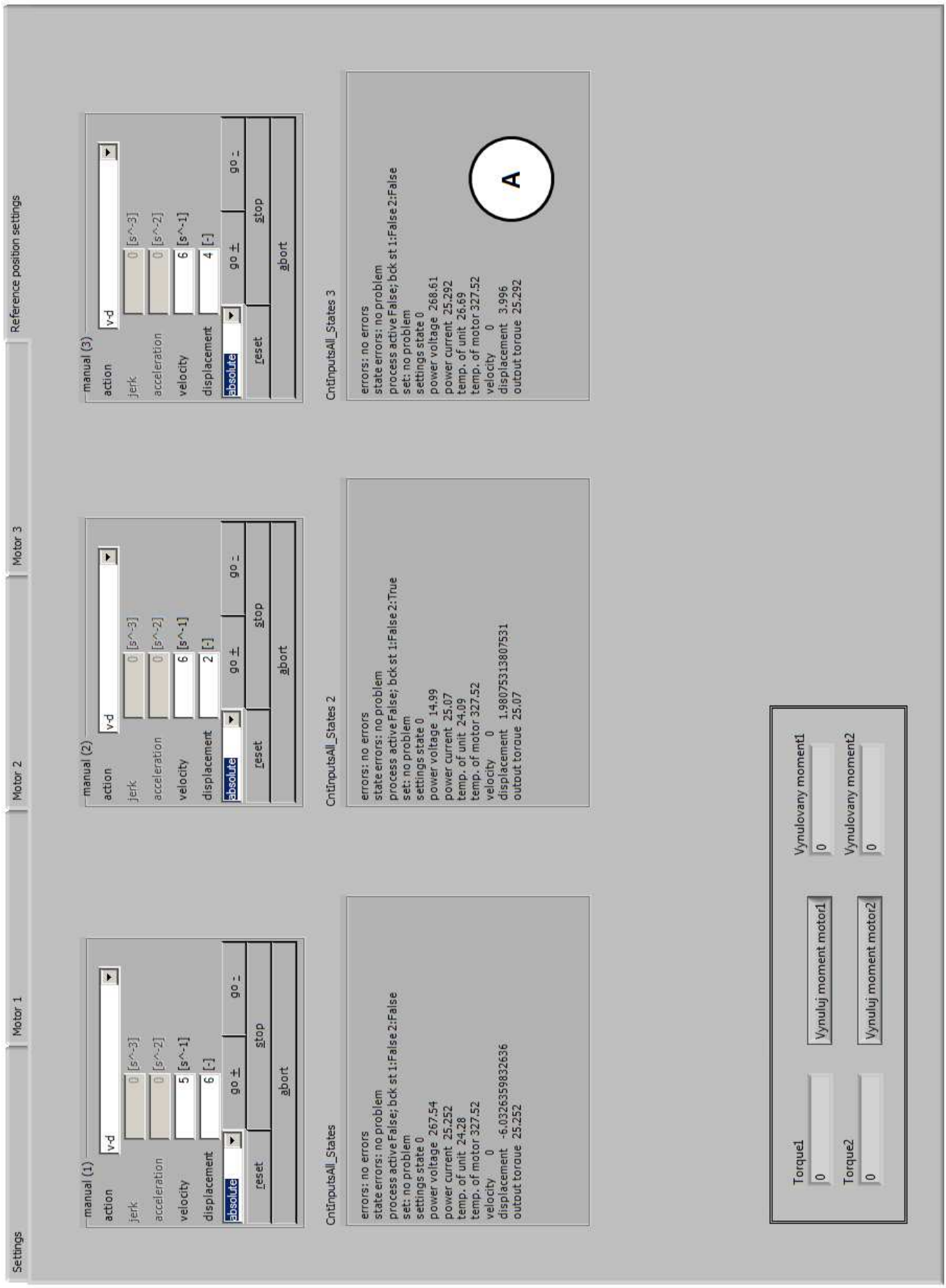
- Elektronická podoba této diplomové práce (Text/TextDP_Hejc.pdf),
- Zdrojový kód vytvořeného software (Software_zdrojovy_kod/BoneBreaker/p_bones.lvproj),
- Spustitelný software (Software_zkompilovano_Win32/BoneBreaker/Bone Breaker.exe),
- NI LabVIEW Runtime (Runtime/LabVIEW_8.6_Runtime_Engine.exe),
- Textový soubor s podmínkami pro spuštění vytvořeného software (Instrukce.txt).

# Příloha č. 1 – Uživatelské rozhraní řídicího software záložka „settings“



**Příloha č. 2 – Uživatelské rozhraní řídicího software záložka „motor“**

**Příloha č. 3 – Uživatelské rozhraní řídicího software záložka „Reference position settings“**



**Příloha č. 4 – Uživatelské rozhraní software pro offline zobrazováni dat záložka „Measured data“**



Measured data | Graphs moments on angles | Graphs moments and angles on times | Graphs moments and angles on times

Path: F:\FlashDisk\zaloha_flasky 19.03.09.23.59\test_data

Deskription: testovani na paternim prvku

Name: Tomas Hejc

Cycle count: 1000,0

Start/Pause time: 14:21:53,196 1.10.2008

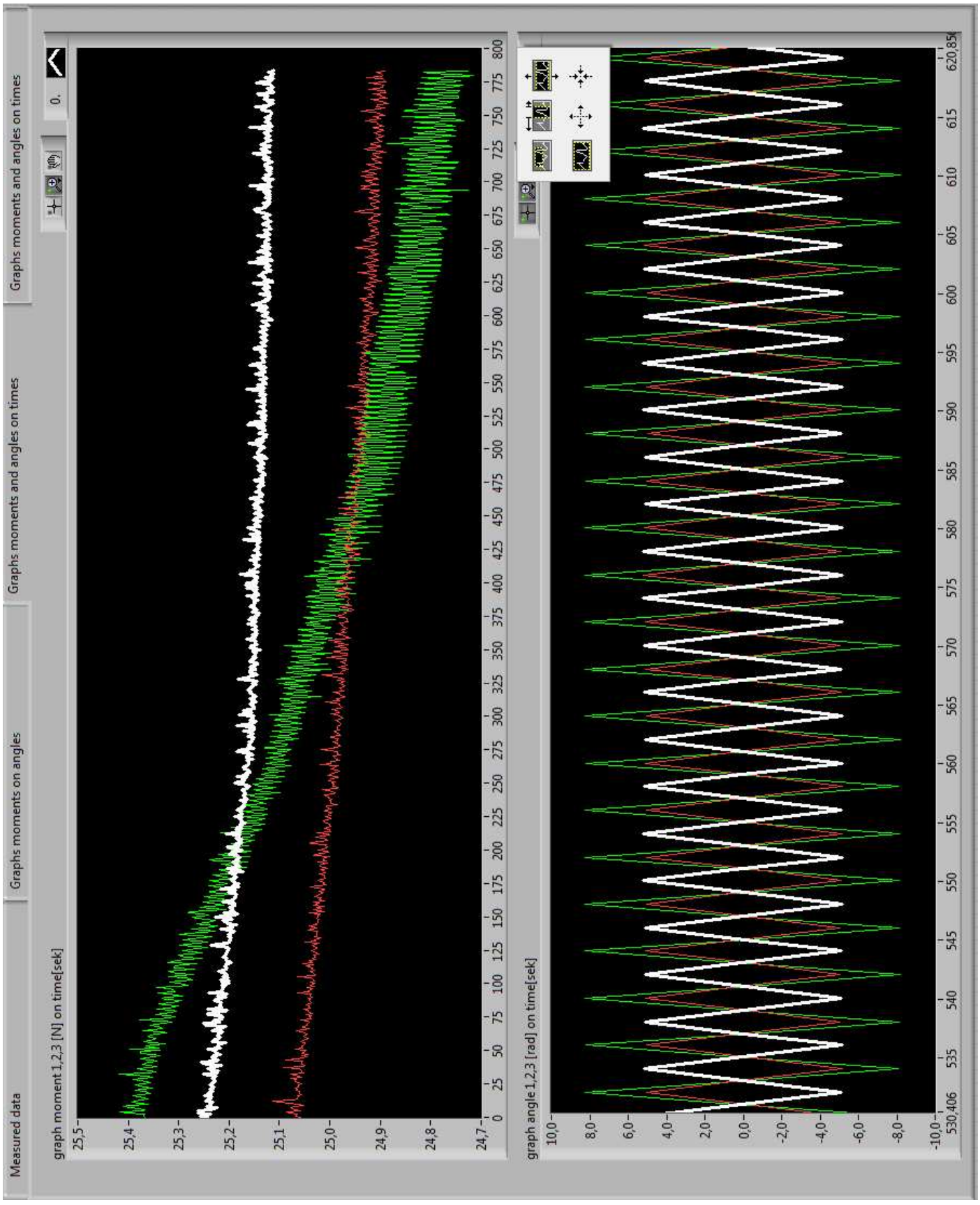
Finysh time: 00:00:00,000 DD.MM.YYYY

TDMS viewer | Exit

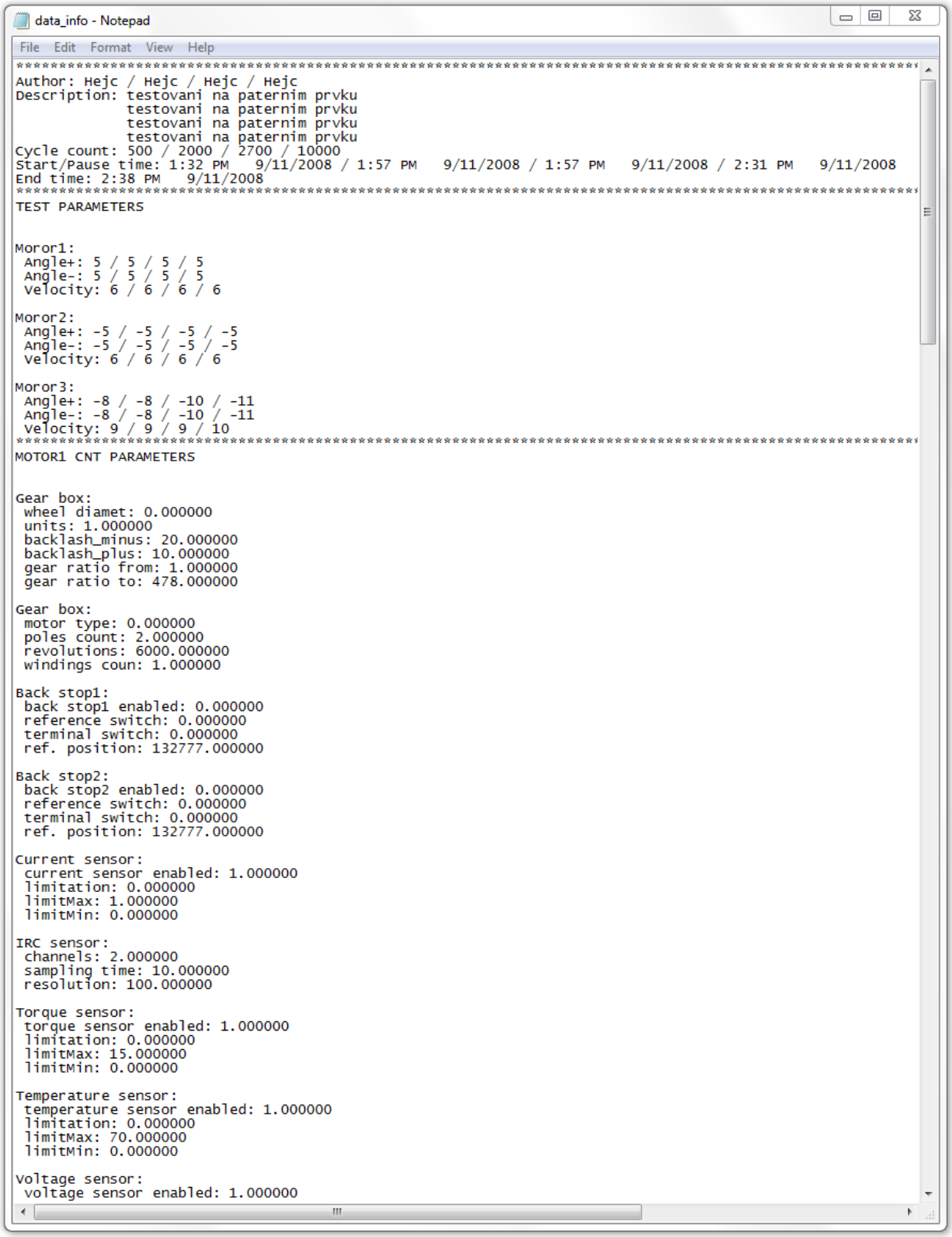
Directory: samples001.tdms

| angle1 | velocity1 | moment1 | angle2 | velocity2 | moment2 | angle3 | velocity | moment3 | time | cycle |
|---|---|---|---|---|---|---|---|---|---|---|
| 0,888703 | 2,259414 | 25,243000 | -0,030126 | -2,259414 | 25,058000 | -0,036000 | -1,800000 | 25,366000 | 0,070101 | 1,000000 |
| 5,158996 | 0,000000 | 25,258000 | -5,076151 | 0,000000 | 25,069000 | -8,271000 | 0,000000 | 25,369000 | 1,071541 | 2,000000 |
| 0,030126 | -5,271967 | 25,236000 | -0,353975 | 5,271967 | 25,064000 | 0,342000 | 9,900000 | 25,368000 | 2,072981 | 2,000000 |
| -5,098745 | 0,753138 | 25,261000 | 4,902929 | 3,765690 | 25,079000 | 8,226000 | 0,000000 | 25,402000 | 3,064406 | 3,000000 |
| -5,000837 | 0,000000 | 25,256000 | 4,985774 | 0,000000 | 25,086000 | 7,803000 | -7,200000 | 25,416000 | 4,065846 | 3,000000 |
| 0,150628 | 3,012552 | 25,261000 | -0,090377 | -5,271967 | 25,059000 | -1,008000 | -8,100000 | 25,370000 | 5,067286 | 4,000000 |
| 5,174059 | 0,753138 | 25,239000 | -5,113808 | 0,000000 | 25,068000 | -8,226000 | 0,000000 | 25,366000 | 6,068727 | 4,000000 |
| 0,045188 | -4,518828 | 25,224000 | -0,045188 | 4,518828 | 25,078000 | 0,387000 | 10,800000 | 25,393000 | 7,070167 | 5,000000 |
| -5,143933 | 0,000000 | 25,244000 | 5,113808 | 0,753138 | 25,074000 | 8,226000 | 0,000000 | 25,409000 | 8,071607 | 5,000000 |
| 0,052720 | 4,518828 | 25,236000 | 0,143096 | -3,012552 | 25,052000 | -0,567000 | -9,000000 | 25,399000 | 9,073047 | 6,000000 |
| 5,166527 | 1,506276 | 25,238000 | -5,211715 | 0,000000 | 25,063000 | -8,226000 | 0,000000 | 25,361000 | 10,064472 | 6,000000 |
| -0,007531 | -5,271967 | 25,231000 | -0,158159 | 6,778243 | 25,097000 | 0,387000 | 9,900000 | 25,392000 | 11,065912 | 7,000000 |
| -5,128870 | 0,000000 | 25,246000 | 5,113808 | 7,531381 | 25,113000 | 8,244000 | 0,000000 | 25,388000 | 12,067352 | 7,000000 |
| 0,052720 | 4,518828 | 25,242000 | 0,112971 | -5,271967 | 25,073000 | -0,531000 | -8,100000 | 25,375000 | 13,068792 | 8,000000 |
| 5,158996 | 0,753138 | 25,239000 | -5,091213 | -4,518828 | 25,064000 | -8,226000 | 0,000000 | 25,360000 | 14,070232 | 8,000000 |
| -0,067782 | -6,025105 | 25,234000 | -0,128033 | 7,531381 | 25,063000 | 0,279000 | 9,000000 | 25,377000 | 15,071672 | 9,000000 |
| -5,128870 | 0,753138 | 25,250000 | 5,128870 | 0,753138 | 25,064000 | 8,253000 | 0,000000 | 25,390000 | 16,073112 | 9,000000 |
| -0,030126 | 5,271967 | 25,237000 | 0,105439 | -1,506276 | 25,056000 | -0,441000 | -9,900000 | 25,375000 | 17,064538 | 10,000000 |
| 5,106276 | 0,753138 | 25,243000 | -5,196653 | -3,765690 | 25,061000 | -8,226000 | 0,000000 | 25,357000 | 18,065978 | 10,000000 |
| -0,022594 | -5,271967 | 25,231000 | -0,173222 | 4,518828 | 25,061000 | 0,288000 | 9,000000 | 25,367000 | 19,067418 | 11,000000 |
| -5,106276 | 0,000000 | 25,234000 | 5,061088 | 4,518828 | 25,067000 | 8,226000 | 0,000000 | 25,386000 | 20,068858 | 11,000000 |
| 0,000000 | 6,025105 | 25,239000 | 0,105439 | -5,271967 | 25,064000 | -0,549000 | -8,100000 | 25,375000 | 21,070298 | 12,000000 |
| 5,151464 | 1,506276 | 25,232000 | -5,113808 | -1,506276 | 25,069000 | -8,244000 | 0,000000 | 25,353000 | 22,071738 | 12,000000 |
| 0,022594 | -5,271967 | 25,229000 | -0,075314 | 4,518828 | 25,062000 | 0,378000 | 9,900000 | 25,367000 | 23,073178 | 13,000000 |
| -5,106276 | 0,753138 | 25,239000 | 5,068619 | 1,506276 | 25,062000 | 8,244000 | 0,000000 | 25,386000 | 24,064603 | 13,000000 |
| 0,007531 | 5,271967 | 25,233000 | 0,015063 | -5,271967 | 25,055000 | -0,531000 | -8,100000 | 25,373000 | 25,066043 | 14,000000 |
| 5,151464 | 0,753138 | 25,234000 | -5,113808 | 0,753138 | 25,063000 | -8,253000 | 0,000000 | 25,356000 | 26,067483 | 14,000000 |
| -0,052720 | -7,531381 | 25,228000 | -0,105439 | 3,765690 | 25,069000 | 0,369000 | 10,800000 | 25,362000 | 27,068923 | 15,000000 |
| -5,158996 | 0,753138 | 25,229000 | 5,174059 | 0,753138 | 25,067000 | 8,235000 | 0,000000 | 25,390000 | 28,070363 | 15,000000 |
| 0,045188 | 3,765690 | 25,234000 | 0,067782 | -8,284519 | 25,077000 | -0,540000 | -9,000000 | 25,371000 | 29,071803 | 16,000000 |
| 5,211715 | 0,753138 | 25,232000 | -5,136402 | -8,284519 | 25,057000 | -8,226000 | 0,000000 | 25,352000 | 30,063229 | 16,000000 |
| 0,037657 | -4,518828 | 25,226000 | -0,173222 | 6,025105 | 25,062000 | 0,387000 | 9,900000 | 25,362000 | 31,064669 | 17,000000 |
| -5,106276 | 0,753138 | 25,252000 | 5,106276 | 5,271967 | 25,077000 | 8,244000 | 0,000000 | 25,419000 | 32,066109 | 17,000000 |
| 0,015063 | 3,765690 | 25,223000 | 0,135565 | -6,025105 | 25,051000 | -0,531000 | -9,900000 | 25,362000 | 33,067549 | 18,000000 |
| 5,091213 | 0,753138 | 25,231000 | -5,106276 | -2,259414 | 25,057000 | -8,217000 | 0,000000 | 25,345000 | 34,068989 | 18,000000 |
| -0,082845 | -5,271967 | 25,221000 | -0,090377 | 4,518828 | 25,057000 | 0,297000 | 9,000000 | 25,357000 | 35,070429 | 19,000000 |
| -5,121339 | 0,753138 | 25,242000 | 5,158996 | 2,259414 | 25,060000 | 8,226000 | 0,000000 | 25,382000 | 36,071869 | 19,000000 |
| 0,030126 | 7,759414 | 25,236000 | 0,143096 | -4,518828 | 25,092000 | -0,459000 | -9,000000 | 25,360000 | 37,073309 | 20,000000 |

**Příloha č. 5 – Uživatelské rozhraní software pro offline zobrazováni dat záložka „Graphs moments and angles on times“**



# Příloha č. 6 – Struktura převedených dat do formátu *.txt

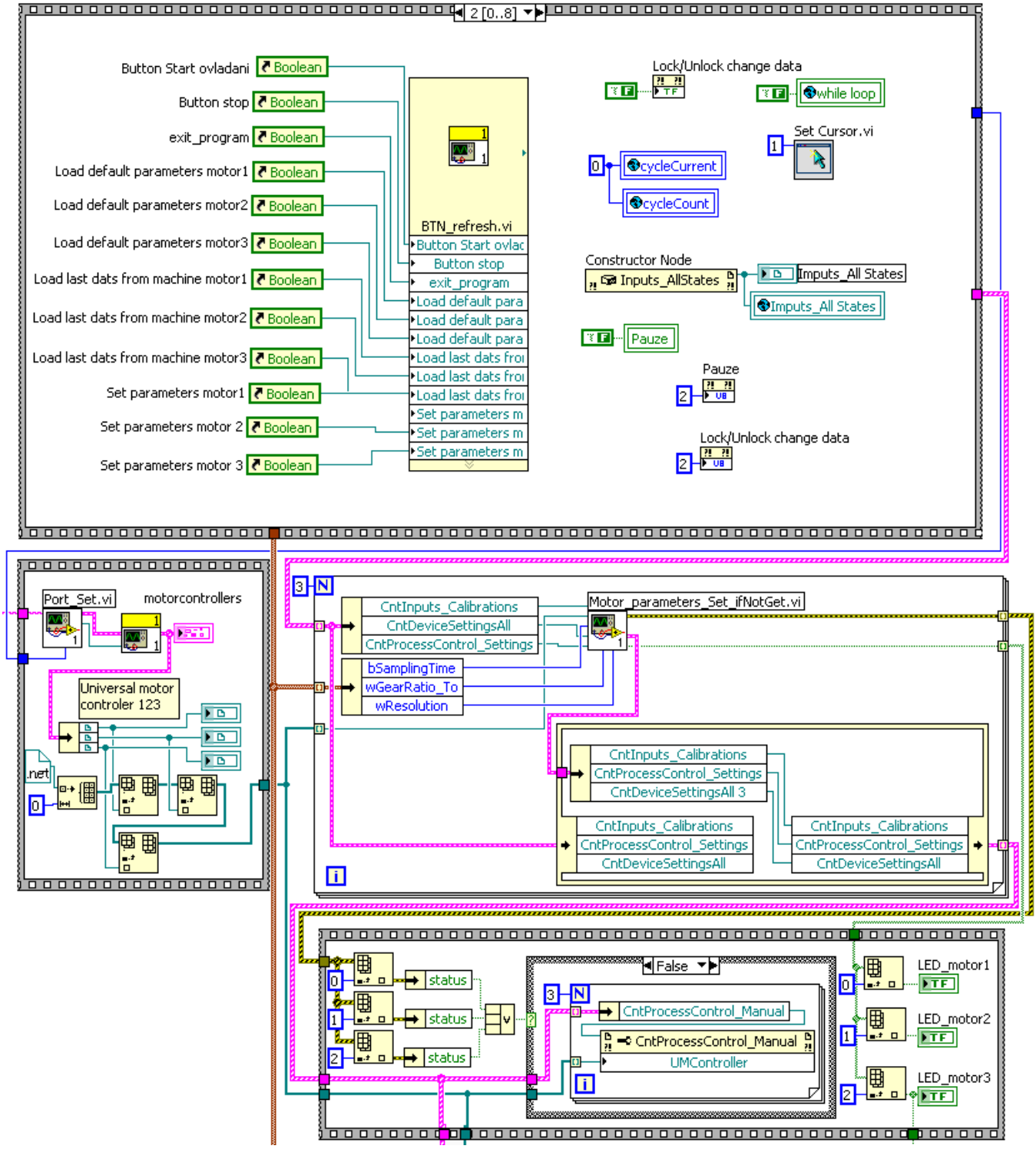


data_info - Notepad

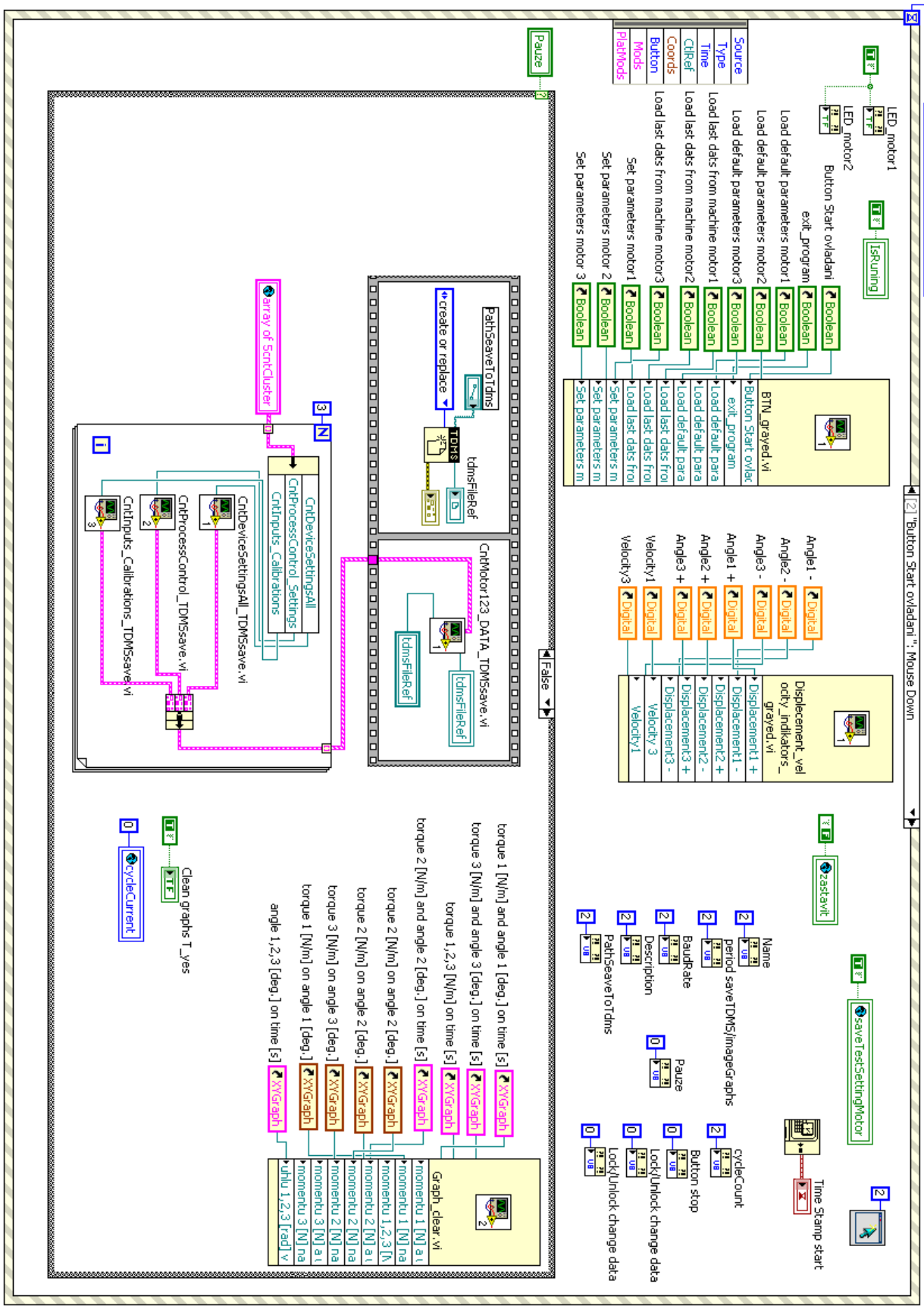
File Edit Format View Help

```
**********************************************************************************************************************
Author: Hejc / Hejc / Hejc / Hejc
Description: testovani na paternim prvku
             testovani na paternim prvku
             testovani na paternim prvku
             testovani na paternim prvku
Cycle count: 500 / 2000 / 2700 / 10000
Start/Pause time: 1:32 PM   9/11/2008 / 1:57 PM   9/11/2008 / 1:57 PM   9/11/2008 / 2:31 PM   9/11/2008
End time: 2:38 PM   9/11/2008
**********************************************************************************************************************
TEST PARAMETERS


Moror1:
 Angle+: 5 / 5 / 5 / 5
 Angle-: 5 / 5 / 5 / 5
 Velocity: 6 / 6 / 6 / 6

Moror2:
 Angle+: -5 / -5 / -5 / -5
 Angle-: -5 / -5 / -5 / -5
 Velocity: 6 / 6 / 6 / 6

Moror3:
 Angle+: -8 / -8 / -10 / -11
 Angle-: -8 / -8 / -10 / -11
 Velocity: 9 / 9 / 9 / 10
**********************************************************************************************************************
MOTOR1 CNT PARAMETERS


Gear box:
 wheel diamet: 0.000000
 units: 1.000000
 backlash_minus: 20.000000
 backlash_plus: 10.000000
 gear ratio from: 1.000000
 gear ratio to: 478.000000

Gear box:
 motor type: 0.000000
 poles count: 2.000000
 revolutions: 6000.000000
 windings coun: 1.000000

Back stop1:
 back stop1 enabled: 0.000000
 reference switch: 0.000000
 terminal switch: 0.000000
 ref. position: 132777.000000

Back stop2:
 back stop2 enabled: 0.000000
 reference switch: 0.000000
 terminal switch: 0.000000
 ref. position: 132777.000000

Current sensor:
 current sensor enabled: 1.000000
 limitation: 0.000000
 limitMax: 1.000000
 limitMin: 0.000000

IRC sensor:
 channels: 2.000000
 sampling time: 10.000000
 resolution: 100.000000

Torque sensor:
 torque sensor enabled: 1.000000
 limitation: 0.000000
 limitMax: 15.000000
 limitMin: 0.000000

Temperature sensor:
 temperature sensor enabled: 1.000000
 limitation: 0.000000
 limitMax: 70.000000
 limitMin: 0.000000

Voltage sensor:
 voltage sensor enabled: 1.000000
```

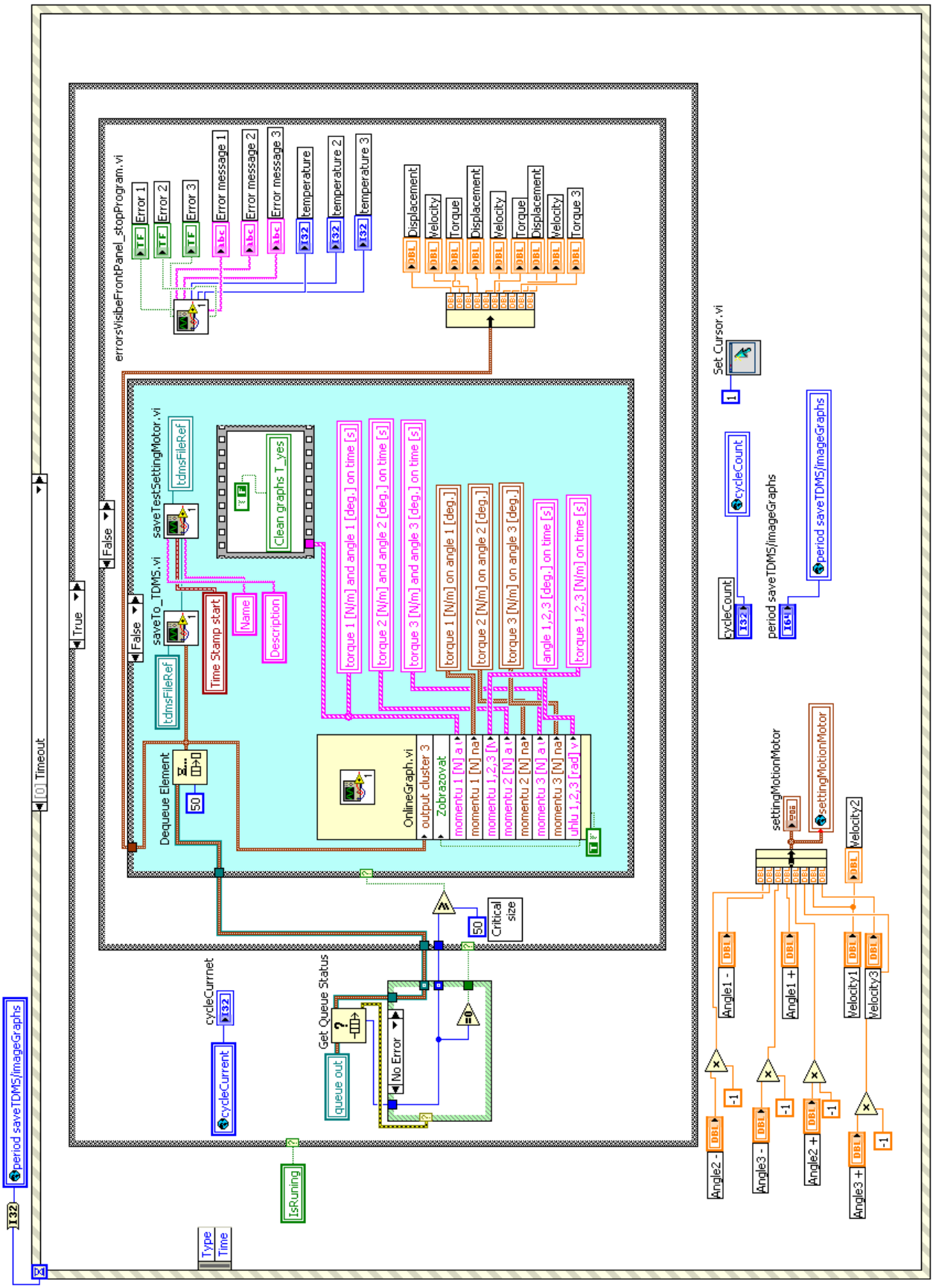
# Příloha č. 7 – Blokové schéma inicializační části řídicího software



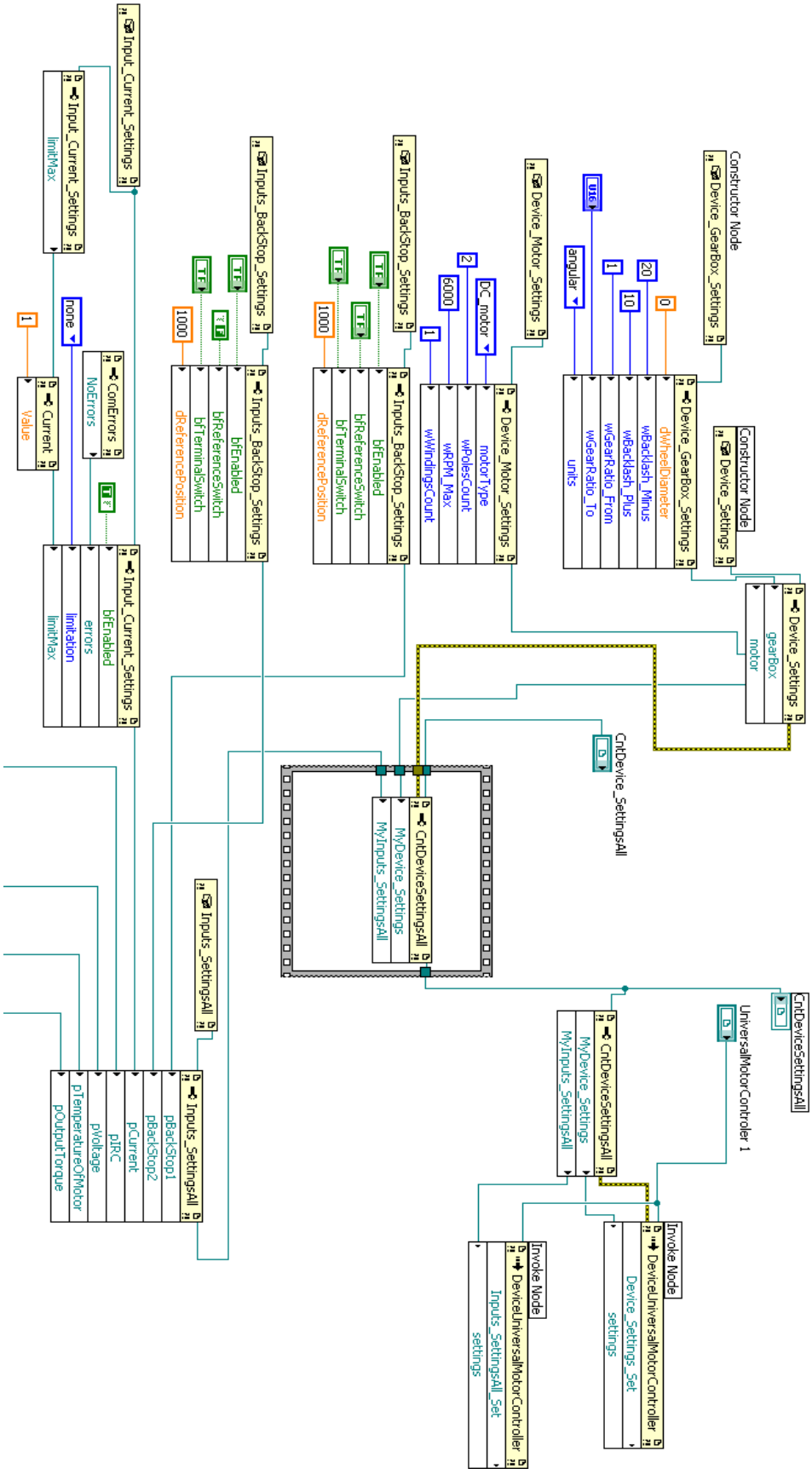
**Příloha č. 8 – Blokové schéma programu prováděné při vzniku události „button start: mouse down“**



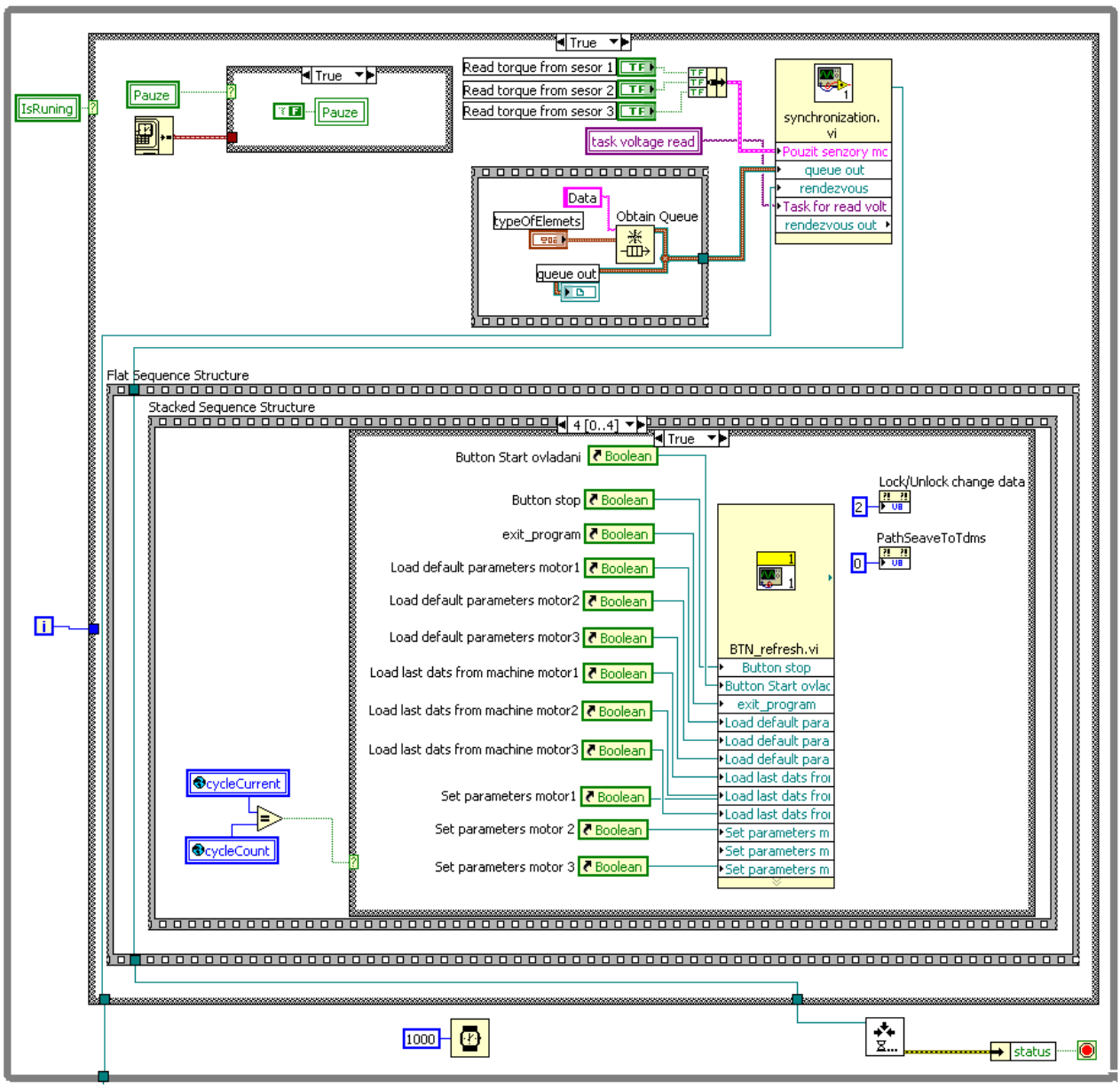
**Příloha č. 9 – Blokové schéma programu prováděné při vzniku události „timeout“**



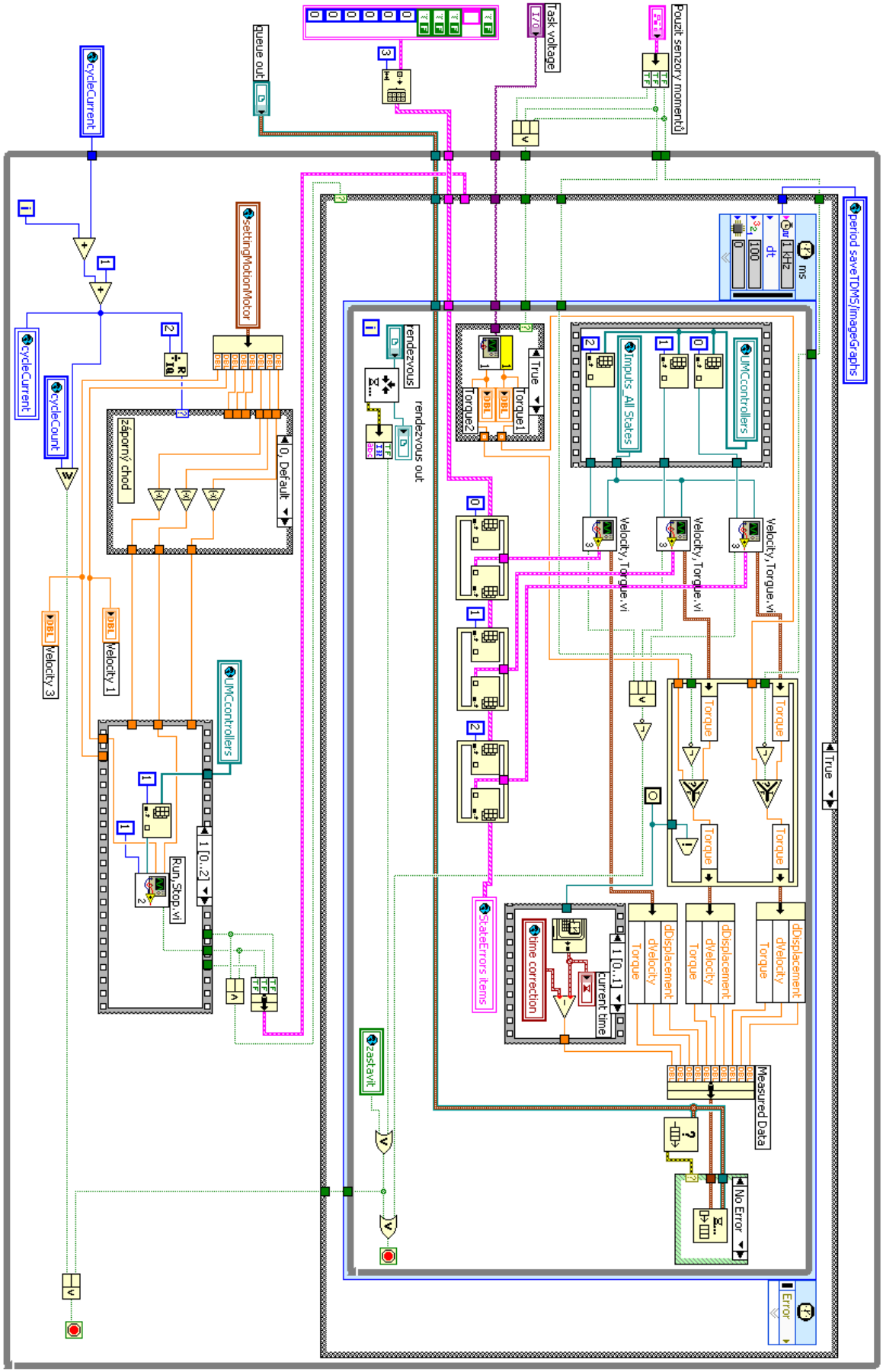
**Příloha č. 10 – Část blokového schématu pro zapsání základních parametrů nastavení souvisejících s pohony do .NET Containers**



**Příloha č. 11 – Blokové schéma části programu zajišťující realizaci chodu pohonů**



**Příloha č. 12 – Blokové schéma SubVI Synchronization obsaženého v části realizace chodu pohonů**



**Příloha č. 13 – Blokové schéma pro načtení uložených dat ve formátu TDMS a jejich následnému rozdělení (do grafů, tabulek a informačních polí)**