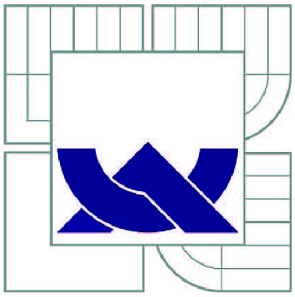
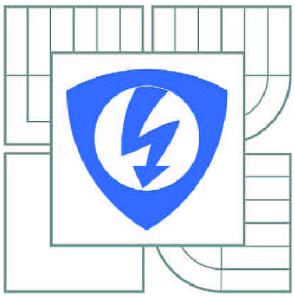


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

DEMONSTRACE PRÁCE IVI OVLADAČE DLE TECHNOLOGIE NATIONAL INSTRUMENTS

DEMONSTRATION IVI DRIVERS NATIONAL INSTRUMENT'S TECHNOLOGY

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

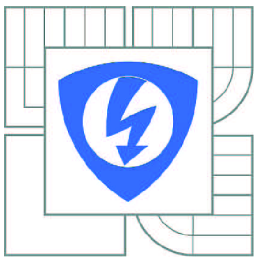
AUTOR PRÁCE — PETR VÁLEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. MILOSLAV ČEJKA, CSc.

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Petr Válek	***ID:***	72777
Ročník:	3	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Demonstrace práce IVI ovladače dle technologie National Instruments

POKYNY PRO VYPRACOVÁNÍ:

1/ Seznamte se s technologií IVI ovladačů měřicích přístrojů dle technologie National Instruments
2/ Navrhněte SW prostředek, který bude dokumentovat (nejlépe graficky) činnost Vámi vybraného (vygenerovaného) ovladače a jeho spolupráci s IVI Engine
3/ SW řešení realizujte, odlaďte a vyzkoušejte, Dokumentujte na něm uváděné výhody IVI ovladačů. Zpracujte dokumentaci k jeho funkci a použití.
4/ Zhodnoťe dosažené výsledky, porovnejte vlastnosti vývojových nástrojů National Instruments pro IVI ovladače
.

DOPORUČENÁ LITERATURA:

.Natinal Instruments.LabWindows/CVI:Instrument Driver Developers Guide.October 2008, P/N 370699C-01.

Termín zadání:	8.2.2010	***Termín odevzdání:***	31.5.2010

Vedoucí práce: Ing. Miloslav Čejka, CSc.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce pojednává o IVI ovladačích, a sice jejich výhodami a vznikem. Dále pak o programu, který demonstruje práci IVI Enginu, což bylo součástí zadání. Popsány jsou zde vlastnosti, funkce a algoritmy programu. Je zde rovněž popsán způsob práce s atributy (přidávání, změna a odstranění) v Labwindows/CVI, stručné pojednání o IVI Foundation a popis ovladače digitálního multimetru vytvořeného k demonstračním účelům. Jsou zde rovněž uvedeny výsledky a postupy kontrolních měření, kterými byly ověřovány uváděné výhody IVI ovladačů. Na závěr jsou uvedeny dosažené výsledky a možná vylepšení.

KLÍČOVÁ SLOVA

IVI ovladače, Labwindows/CVI, National Instruments

ABSTRACT

This thesis treats IVI drivers, their advantages and origins. It describes program demonstrating IVI Engine work, which was part of the assignment. Furthermore are described program properties, functions and algorithms. There is also described way of working with attributes (adding, editing and deletion) in Labwindows/CVI, brief chapter about IVI Foundation and description of digital multimeter driver made for demonstration purposes. Also, results and control measurement procedures, with which presented IVI drivers features were checked, were described. Lastly, achieved results and possible enhacements are presented.

KEYWORDS

IVI drivers, Labwindows/CVI, National Instruments

VÁLEK P.: *Demonstrace práce IVI ovladače dle technologie National Instruments.*
Bakalářská práce. FEKT VUT v Brně, 2010,44s., 12s. příloh

PROHLÁŠENÍ

„Prohlašuji, že svou bakalářskou práci na téma „Demonstrace práce IVI ovladače dle technologie National Instruments“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

V Brně dne: **28. května 2010** ……
podpis autora

PODĚKOVÁNÍ

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

V Brně dne: **28. května 2010** ……
podpis autora

OBSAH

1. ÚVOD 8
2. IVI 9
2.1 Úvod 9
2.2 IVI Foundation 9
2.3 IVI ovladače 10
2.3.1 IVI Třídy 10
2.4 Koncepce vybraného řešení 10
3. POPIS PROGRAMU 13
3.1 Ovládání 13
3.2 Hlavní panel 13
3.3 Panel třídy digitální multimetr 14
3.4 Funkce 15
3.4.1 Funkce main 15
3.4.2 Funkce MeasureProceed 16
3.4.3 Callback funkce prvků panelu 17
3.5 Ovladač multimetru 17
3.6 Funkce IVI ovladače 17
3.6.1 Funkce InitWithOptions 18
3.6.2 Funkce ConfigureMeasurement 20
3.6.3 ConfigureTrigger 21
3.6.4 Funkce Read 21
3.6.5 Funkce Close 23
3.6.6 Funkce SelfTest 23
3.7 Simulovaný hardware multimetru 24
4. MODIFIKACE ATRIBUTŮ 27
4.1 Přidání atributu 29
4.2 Odstranění atributu 31
5. PROVEDENÉ MĚŘENÍ 33

5.1 Ukládání stavu 33
5.2 Volitelná kontrola rozsahů 35
5.3 Volitelný dotaz na stav přístroje 35
5.4 Simulace přístroje 36
5.5 Kontrola IVI ovladače 36
5.5.1 Behavior test 37
5.5.2 Compliance test 38
5.5.3 Structure test 38
5.5.4 Function Panel test 40
6. ZÁVĚR 41

SEZNAM OBRÁZKŮ

Obrázek 2.1 Modifikovaná hierarchie IVI 12
Obrázek 3.1 Okno hlavního panelu 13
Obrázek 3.2 Okno panelu třídy digitální multimetr 14
Obrázek 3.3 Algoritmus funkce main 15
Obrázek 3.4 Algoritmus funkce MeasureProceed 16
Obrázek 3.5 Algoritmus funkce InitWithOptions 19
Obrázek 3.6 Algoritmus funkce ConfigureMeasurement 20
Obrázek 3.7 Algoritmus funkce ConfigureTrigger 21
Obrázek 4.1 Okno editoru atributů 29
Obrázek 4.2 Okno přidání atributu 30
Obrázek 4.3 Okno mazání nepotřebných funkcí 31

SEZNAM TABULEK

Tabulka 1 Přenesená data s a bez state cachingu 34
Tabulka 2 Textový výstup měření v režimu IVI simulace 36

1. ÚVOD

Zadáním práce je vypracování simulačního programu, který bude demonstrovat práci IVI Enginu. Tento bude vypracován ve vývojovém prostředí od National Instruments LabWindows/CVI [3]. Program bude řízen událostmi a jeho součástí je simulační IVI ovladač pro fiktivní multimetr. Ovladač bude obsahovat pouze některé základní funkce, neboť naprogramovat plnohodnotný ovladač není cílem práce. Demonstrace bude znázorněna textovým záznamem a zobrazením volaných callback funkcí, simulované SCPI[1] komunikace, zápisem a čtením atributů přístroje. V závěru práce pak budou uvedeny zjištěné výsledky ověření udávaných výhod IVI ovladačů. Výhody IVI ovladačů budou vybrány z dokumentace National Instruments [5] a ovladač bude otestován pro IVI kompatibilitu.

[1] SCPI – Standard Commands for Programmable Instrumentation, sada ASCII příkazů pro komunikaci s přístroji [1]

2. IVI

2.1 ÚVOD

IVI (Interchangeable Virtual Instrument) je souhrn specifikací pro programování přístrojových ovladačů. Vznikl v roce 1998, kdy se spojilo několik koncových uživatelů, výrobců hardwaru a programátoru softwaru v organizaci IVI Foundation. Práce s IVI kompatibilními zařízeními má několik nesporných výhod [1]:

- Konzistence – všechny IVI ovladače mají jednotný model jak přistupovat k funkcím zařízení, programátor se tedy nemusí pokaždé znova učit.
- Zaměnitelnost – uživatel má možnost vyměnit přístroj za jiný model s minimální editací kódu.
- Simulace – ovladače jsou schopny simulovat i fyzicky nedostupný přístroj, čímž usnadňují testování v laboratořích.
- Ověřování rozsahu (Range Checking) – uživatel má zaručeno, že zařízení nepošle požadavek mimo rozsah přístroje.
- Ukládání stavu (State Caching) – ovladač si ukládá stav přístroje, a odstraňuje redundantní komunikaci. Tím může dosáhnout viditelného urychlení měření.

2.2 IVI FOUNDATION

Konsorcium IVI Foundation se zabývá vyvíjením specifikací pro programování přístrojových ovladačů, s výhodami uvedenými výše. Neformální setkání zakládajících členů se konala již od července 1997, avšak veřejnosti se představili až na konferenci National Instruments Week 1998. Mezi zakládající členy patřili firmy [4]: Anritsu, Advantest, Ascor The Boeing Copany, GenRad, Hewlett-Packard, Keithley, LeCroy, Lockheed Martin, Lucent Technologies, Marconi Integrated Systems (dříve GDE Systems), Marconi Test systéme, National Instruments, Raytheon TI Systéme, Northrop Grumman ESSD, Racal Instruments, Rohde & Schwarz, Tektronix, Teradyne, TYX Corporation, Vektrex, Wavetek.

2.3 IVI OVLADAČE

2.3.1 IVI Třídy

Ovladače specifikace IVI se dělí do osmi tříd [1], a sice:

- Digitální multimetr
- Osciloskop
- Funkční generátor
- Stejnosměrný zdroj
- Přepínač
- Meřič výkonu
- Spektrální analyzátor
- RF generátor

Každá třída má své základní (base class capabilities) a rozšířené (class extension capabilities) schopnosti. Nezávisle na třídě má ještě každý ovladač vrozené schopnosti (inherent capabilities). Ovladač splňující podmínky některé třídy (class-compliant specific driver) potom má vrozené, základní i rozšířené schopnosti dané třídy (všechny které fyzicky podporuje model přístroje).

2.4 KONCEPCE VYBRANÉHO ŘEŠENÍ

Způsob jakým bude možno ověřit funkčnost National Instruments IVI Enginu je naznačen na obrázku 2.1. Jedná se o modifikované schéma hierarchie měřícího systému s IVI technologií [1]. Modifikace spočívá v odchycení komunikace mezi VISA knihovnou a hardwarem přístroje, a její odklonění do příslušné simulační části programu. Takto lze získat přehled o dění uvnitř systému, narozdíl od použití simulace na úrovni IVI ovladače, kdy nedojde k volání funkcí VISA knihovny a tedy i pokusu o komunikaci s hardwarem.

Popis jednotlivých vrstev :

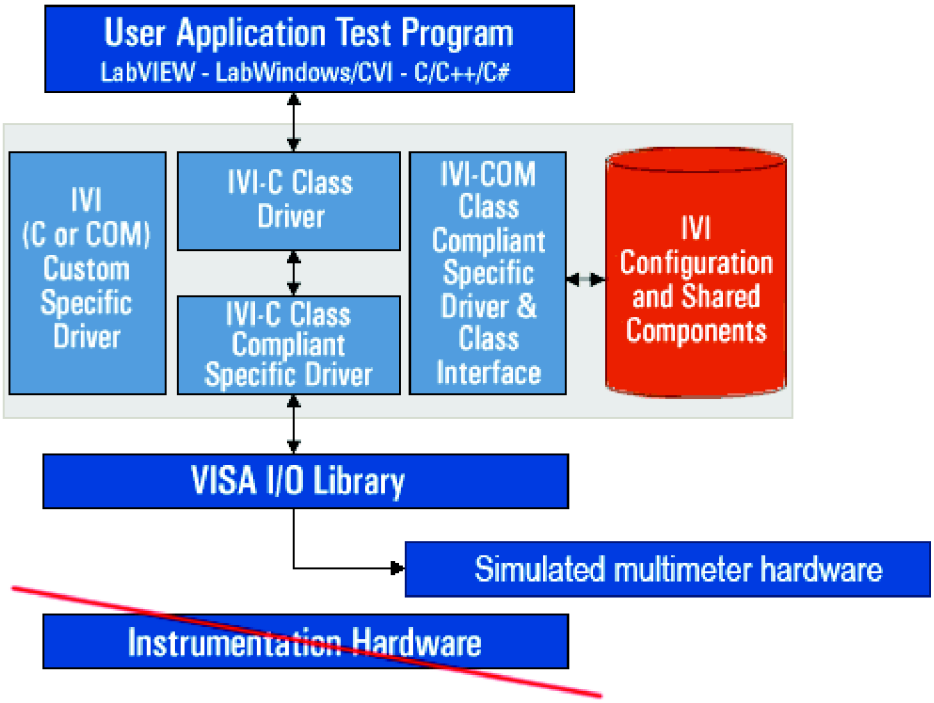
- User Application Test Program - V této nejvyšší vrstvě se nachází program uživatelského rozhraní (hlavní panel, panel třídy), který dále rozhoduje o komunikaci s IVI ovladačem.

- IVI část - do tohoto bloku spadá jednak IVI ovladač a IVI Engine, který je zabudován do LabWindows. V této vrstvě se nachází vygenerovaný ovladač multimetru použitý v této práci.

- VISA I/O Library - VISA vstupně výstupní knihovna, která zajišťuje komunikaci mezi IVI systémem a samotným hardwarem. Některé její funkce jsou v této práci nahrazeny tak, aby dovedly samy odpovídat na dotaz k přístroji. Dochází zde tedy k určité integraci simulovaného hardwaru do této vrstvy.

- Simulated multimeter hardware - Chování přístroje je naprogramováno a provázáno s voláním VISA funkcí, zároveň je schopen uložení některých atributů (nejedná se o paměť atributů IVI Enginu, tedy je možné zjistit nesrovnalost atributu v IVI Engine paměti a paměti simulovaného hardwaru).

- Instrumentation Hardware - Fyzický hardware přístroje není v této práci použit, ani jeho použití není v rámci použité koncepce možné, protože komunikace VISA knihovny k němu nemá přístup (Je možné jím nahradit simulovaný hardware, ale musí mít stejné vlastnosti jako simulovaný přístroj ve visa_custom.c kódu a tento soubor pak ještě musí být upraven, aby nedocházelo k přerušení VISA volání).



Obrázek 2.1 Modifikovaná hierarchie IVI

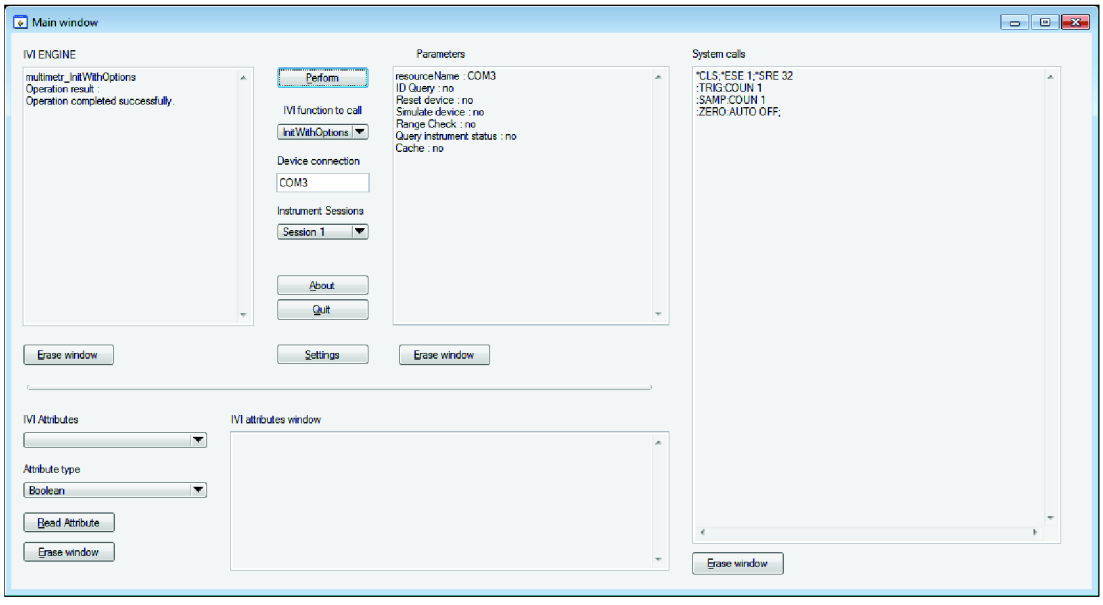
3. POPIS PROGRAMU

Program je vytvořený v prostředí LabWindows/CVI [3] v jazyce C. Skládá se z části hlavního programu (bak.c, panel.h, panel.uir), části IVI ovladače multimetru (multimetr.c, multimetr.h, multimetr.fp, multimetr.sub) a části nasimulovaného hardwaru (visa_custom.c, visa_custom.h). Zdrojové soubory jsou k dispozici na přiloženém médiu. Návod k obsluze vytvořeného programu je v příloze A.

3.1 OVLÁDÁNÍ

Program se ovládá pomocí tlačítek na hlavním a třídovém panelu. Na hlavním panelu (Obrázek 3.1) uživatel zvolí funkci, kterou chce volat, v třídovém panelu (Obrázek 3.2) volí parametry. Program podporuje práci s více simulovanými přístroji najednou. Uživatel volí, se kterým přístrojem pracuje v nabídce **Instrument Sessions**.

3.2 HLAVNÍ PANEL



Obrázek 3.1 Okno hlavního panelu

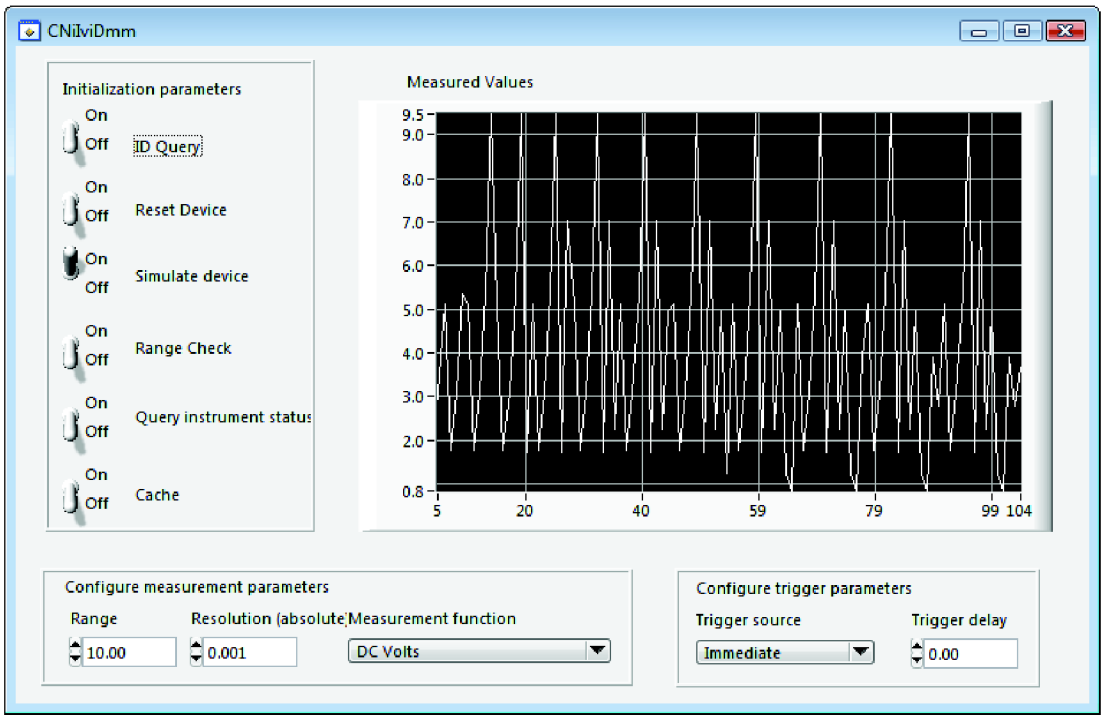
Na hlavním panelu jsou 4 textboxy. **IVI Engine textbox** zaznamenává volání funkcí IVI ovladače a zároveň jeho návratovou hodnotu (převedenou na slovní popis funkcí multimetr_error_message).

V textboxu **Parameters** se zobrazují parametry volaných funkcí.

Textbox **System calls** zobrazuje práci IVI Engine s atributy, volání VISA funkcí (tedy pokus o komunikaci IVI ovladače s přístrojem), žádosti o uzamknutí/odemknutí session[2]. Zaznamenávaná data v textboxu System calls lze nastavit tlačítkem **Settings** (toto nemá efekt na již zaznamenaná data).

Uživatel má dále možnost zobrazit hodnotu libovolného atributu ve spodní části panelu pomocí tlačítek **Read atribute** do textboxu **IVI attributes window**. Výběr atributu se provádí zvolením typu atributu (Boolean, Integer, Real, Vi Session, String), a poté názvu atributu z nabídky IVI Attributes. Obsah kteréhokoliv textboxu je možno vymazat příslušným tlačítkem **Erase**.

3.3 PANEL TŘÍDY DIGITÁLNÍ MULTIMETR



Obrázek 3.2 Okno panelu třídy digitální multimetr

Panel třídy slouží k nastavování parametrů volaných funkcí z hlavního panelu. Část okna **Initialization parameters** nastavuje parametry funkce

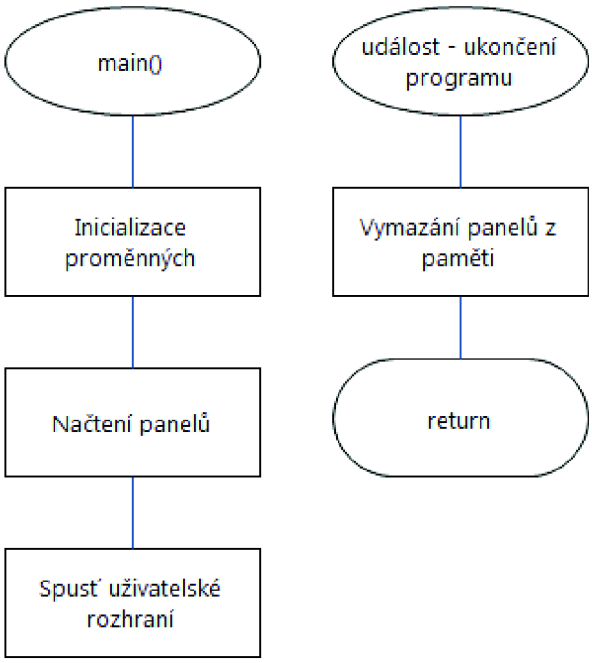
[2] session - jedna relace ovladače, obvykle zahájena inicializační funkcí a ukončena funkcí close

InitWithOptions, a sice ID querying, reset zařízení při inicializaci, simulaci zařízení, kontrolu rozsahu, dotazování se na status přístroje a state caching. V části **Configure measurement parameters** jsou parametry funkce Configure Measurement, konkrétně rozsah, rozlišení a měřící funkce. V části **Configure trigger parameters** potom zdroj trigger signálu a prodlení.

3.4 FUNKCE

3.4.1 Funkce main

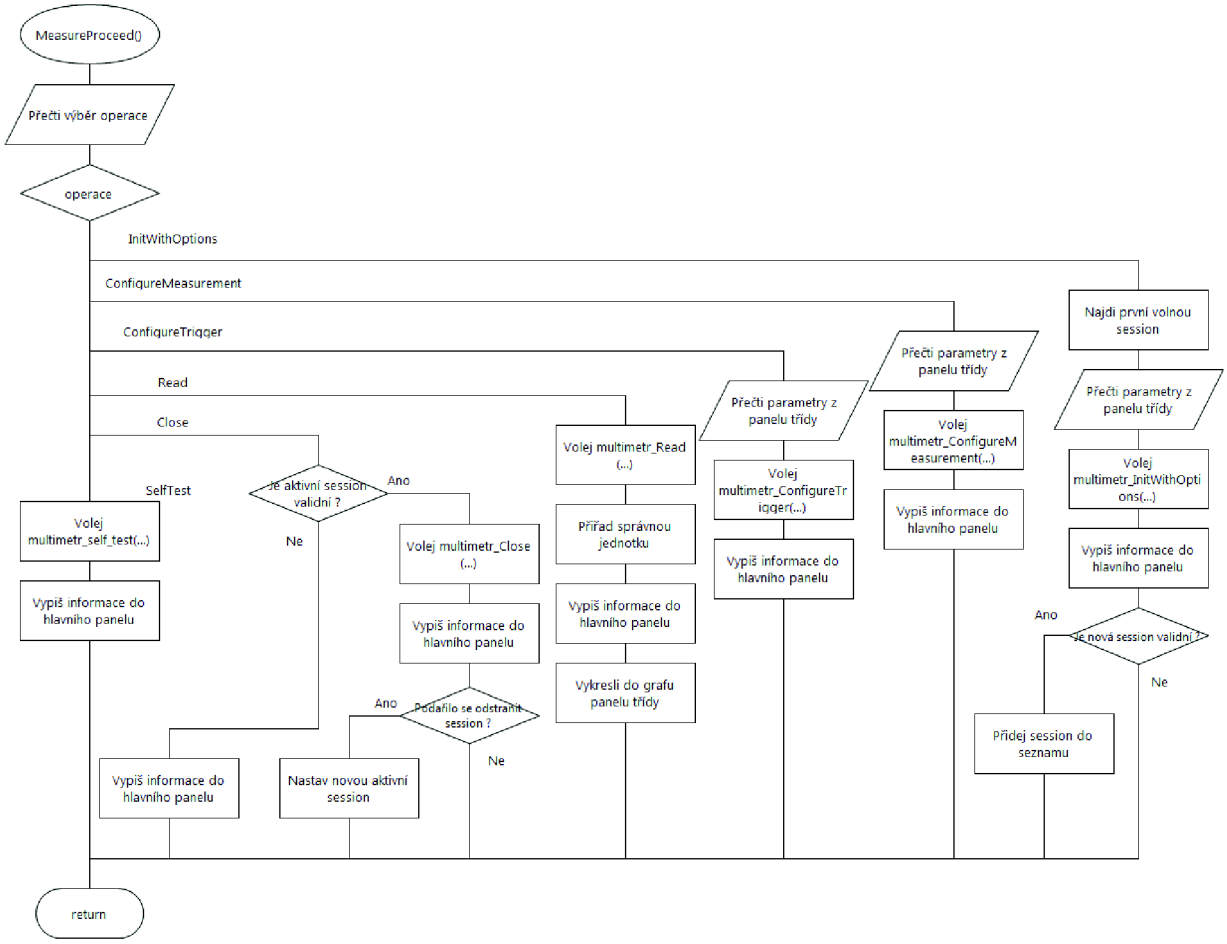
Hlavní funkce programu je relativně jednoduchá, protože se jedná o událostmi řízený program (Po spuštění uživatelského rozhraní je program zacyklen do doby než přijde žádost o ukončení). Inicializace proměnných spočívá ve vynulování přečtených hodnot multimetrem a vynulování řetězce sessions (celkem může být najednou simulováno až 256 přístrojů). Sessions jsou uloženy v řetězci struktur, kde ke každé je přidefinována třída přístroje a zda-li je session nainicializována. Před spuštěním samotného rozhraní jsou načteny uživatelské panely a implicitní hodnoty panelu settings.



Obrázek 3.3 Algoritmus funkce main

3.4.2 Funkce MeasureProceed

Funkce je volána po stisku tlačítka **Perform** a má za úkol provést vybranou operaci ze seznamu IVI function to call. (příloha B.1) Funkce kontroluje, zda-li je vybrána existující session pro odeslání operace. Výstupy do textboxů jsou formátovány voláním funkce IVI ovladače "multimetr_error_message", nejsou tedy pouze v číselné podobě.



Obrázek 3.4 Algoritmus funkce MeasureProceed

3.4.3 Callback funkce prvků panelu

Callback funkce prvků na panelech jsou generované prostředím CVI, a protože jsou téměř totožné, je uveden pro příklad funkce pro tlačítko :

```
int CVICALLBACK CommandButton4CB (int panel, int control, int event,
                    void *callbackData, int eventData1, int eventData2)
{
        switch (event)
        {
                case EVENT_COMMIT:
                        HidePanel (aboutpanel);
                        break;
        }
        return 0;
}
```

Událost EVENT_COMMIT je obecná událost pro aktivaci prvku na panelu (například stisknutí tlačítka, vybrání položky z nabídky apod.). Zde je konkrétně uveden kód tlačítka OK na About panelu, které zavře panel (funkce HidePanel).

3.5 OVLADAČ MULTIMETRU

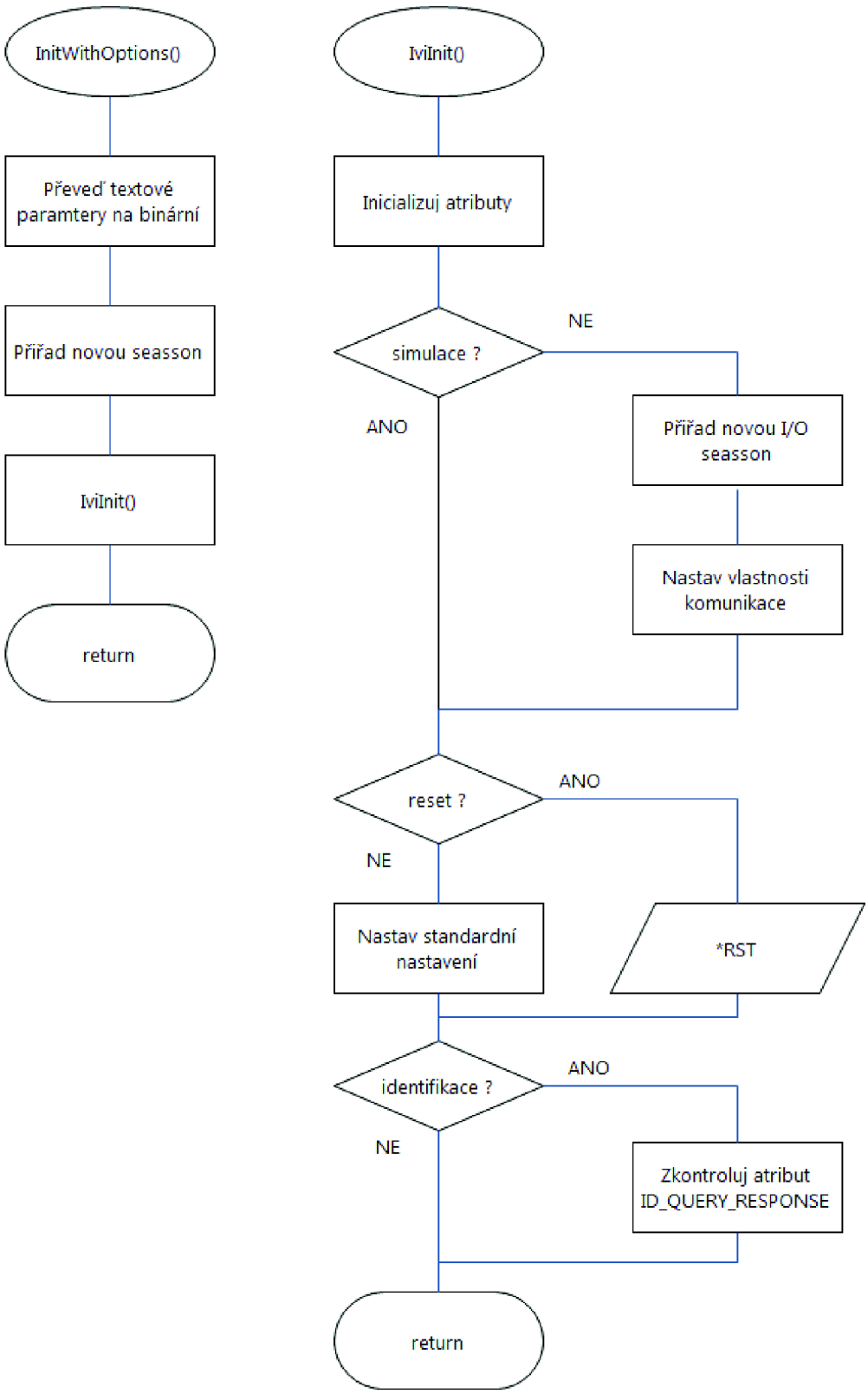
Ovladač byl vygenerován za pomocí IVI Instrument driver wizardu v LabWindows/CVI, nastaveným na implicitní nastavení, tj. GPIB sběrnice, podporující žádost o identifikaci, reset, selftest, dotaz na chybu a dotaz na verzi firmwaru. Byly použity vygenerované příklady, upravené k zobrazovaní dat na hlavním panelu. Přestože přístroj není fyzicky přítomen, a nedochází tedy ke skutečné komunikaci po sběrnici, musí být zadán existující port v počítači, aby došlo ke korektní inicializaci. Veškerá potenciální komunikace směřující na tento port (tedy volání VISA funkcí) je zachycena programem a je na ni odpovězeno simulovaným hardwarem (viz kap. 3.7).

3.6 FUNKCE IVI OVLADAČE

Zde budou rozebrány funkce IVI ovladače volané z hlavního panelu. V rámci zachování přehlednosti nejsou u názvů uvedeny předpony multimetr_ a nejsou zmiňovány kontroly proměnných (např. nulové ukazatele). Tyto funkce byly vygenerovány IVI Instrument driver wizardem, jediné jejich úpravy jsou výpisy volání do textboxů a jsou zde popsány pro pochopení činnosti celého programu.

3.6.1 Funkce InitWithOptions

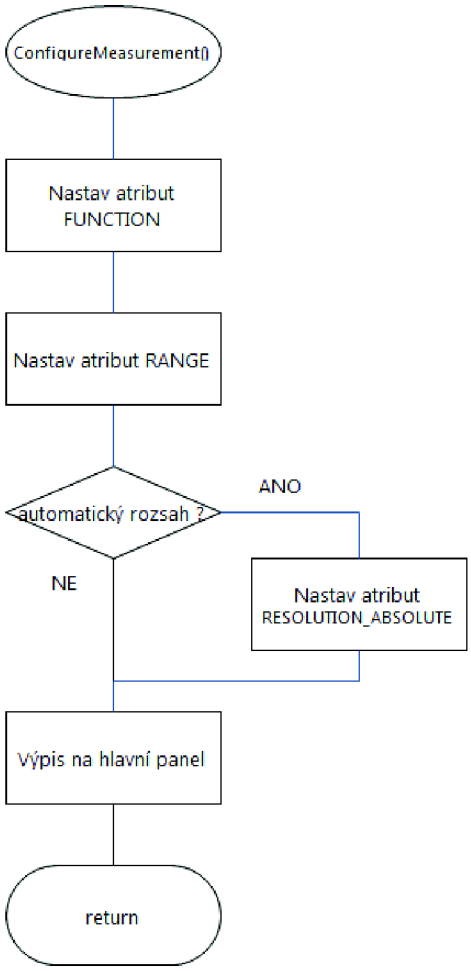
InitWithOptions je funkce inicializační, která má za úkol vytvořit Vi Session přístroje a nadefinovat její vlastnosti (např. zda-li má použít state caching). Vlastnosti jsou předávány za pomocí textových řetězců. Tyto řetězce funkce převede na binární proměnné a volá s nimi funkci **IviInit**. Na počátku této funkce jsou inicializovány atributy přístroje a je-li přístoj fyzicky přítomen (nebo je-li simulován až na nižší vrstvě než v IVI ovladači) je mu přidělena nová vstupně výstupní session a je nakonfigurována komunikace pomocí knihovny VISA. Na konec je zkontrolována žádost o reset či identifikaci a jsou vyslána příslušná data. Je-li při vykonávání funkce zaznamenána chybová návratová hodnota některé z podfunkcí (např. při špatně nastaveném komunikačním portu nebo chybové hodnotě odpovědi na dotaz na stav **ESR?*) není nová session vytvořena.



Obrázek 3.5 Algoritmus funkce InitWithOptions

3.6.2 Funkce ConfigureMeasurement

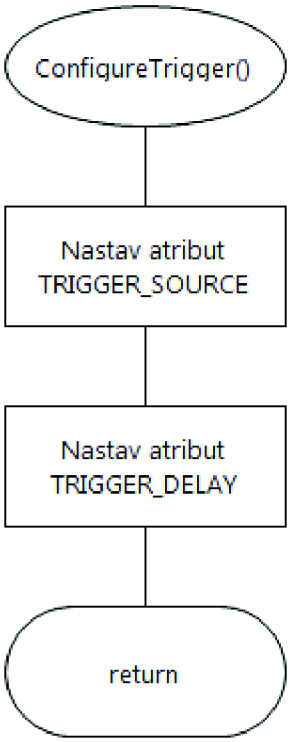
Funkce je volána pro nastavení měření dané session. Přijaté parametry jsou měřící funkce, rozsah a rozlišení. Tyto parametry jsou po zavolání funkce převedeny do atributů **MULTIMETR_ATTR_FUNCTION**, **MULTIMETR_ATTR_RANGE** a **MULTIMETR_ATTR_RESOLUTION_ABSOLUTE**. Hodnota rozlišení je převedena do atributu pouze tehdy, není li rozsah roven hodnotě **MULTIMETR_VAL_AUTO_RANGE_ON** (tedy není-li zapnut automatický rozsah).



Obrázek 3.6 Algoritmus funkce ConfigureMeasurement

3.6.3 ConfigureTrigger

Funkce slouží k nastavování spouštění měření. Předávané parametry jsou zdroj spouštění (ihned, vnější a softwarové spouštění) a prodlení v sekundách. Pracuje s atributy **MULTIMETR_ATTR_TRIGGER_SOURCE** a **MULTIMETR_ATTR_ _TRIGGER_DELAY** (příoha B.2).



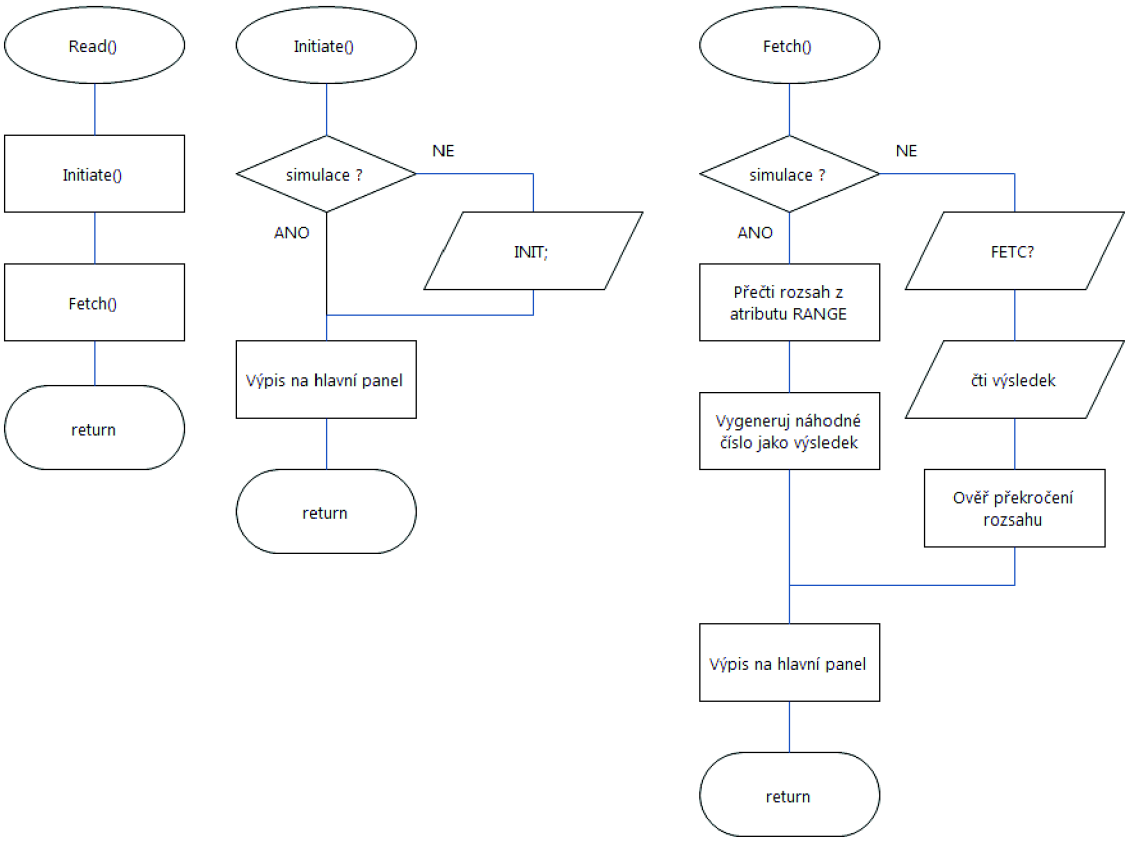
Obrázek 3.7 Algoritmus funkce ConfigureTrigger

3.6.4 Funkce Read[3]

Tato funkce iniciuje měření a vrátí volající funkci výsledek. Parametry funkce jsou VI session měřícího zařízení, limit doby měření (tzv. timeout) a ukazatel na proměnnou, kde bude uložen změřemý výsledek. Jsou volány další dvě funkce – **initiate** a **fetch**. Initiate pracuje pouze s fyzickým přístrojem (případně simulovaným na nejnižší vrstvě IVI hierarchie) , a vyšle mu SCPI příkaz *„INIT;“*. Fetch vyšle

[3] Nejedná se o funkci viRead VISA knihovny ale o Read funkci IVI ovladače (multimetr.c)

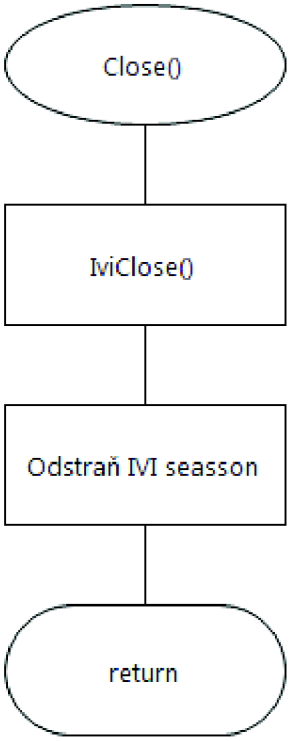
přístroji příkaz *„FETC?“*, čímž jej požádá o vyslání výsledku. Funkce čeká na přijetí výsledku pouze do limitu doby měření (uživatelem nelze měnit, implicitní hodnota je 10 sekund).



Obrázek 3.8 Algoritmus funkce Read

3.6.5 Funkce Close

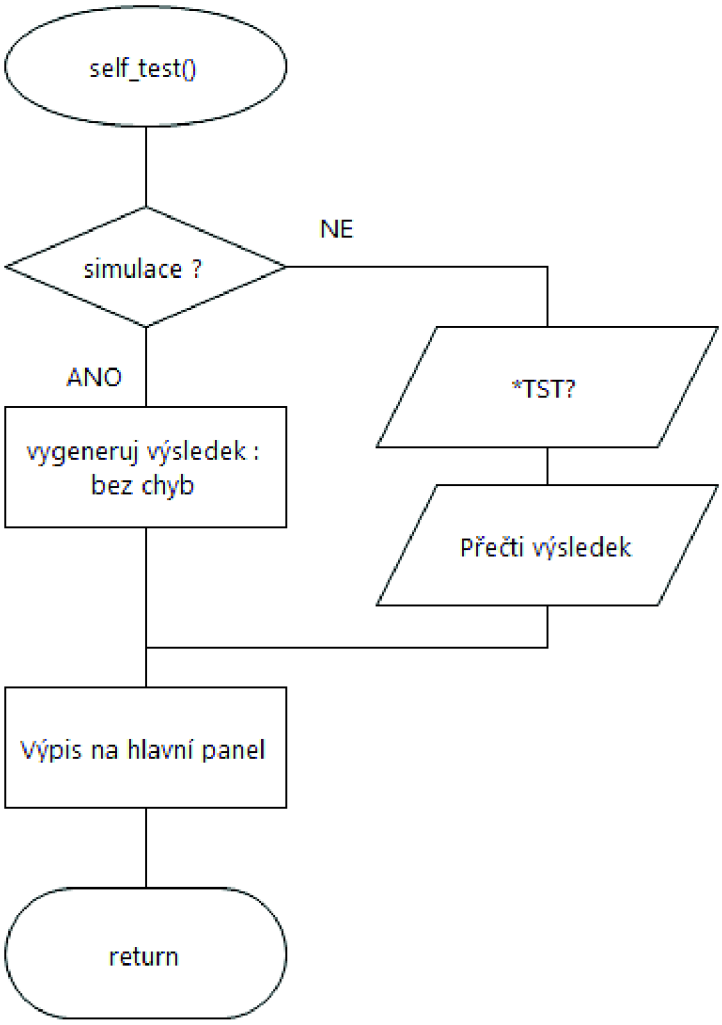
Funkce slouží k ukončení práce s určitou session, obstarává odstranění této session, a jejích proměnných, z paměti. Pracuje s atributem **IVI_ATTR_IO_SESSION**.



Obrázek 3.9 Algoritmus funkce Close

3.6.6 Funkce SelfTest

Tato funkce provede na přístroji diagnostický test. Tento test je iniciován vysláním příkazu *„ *TST?“*. Výsledek je vrácen volající funkci ve formě 16bitového čísla a textového řetězce. V režimu IVI simulace, i při simulaci ve VISA vrstvě, přístroj vždy vrátí výsledek 0 – „No error“. Chceme-li, aby přístroj vracel jiný výsledek, musíme změnit buď soubor multimetr.c, konkrétně funkci multimetr_self_test v režimu IVI simulace, nebo soubor visa_custom.c funkci viCustomScanf v podmínce pro LastPrintCode rovnu 2. Je možné změnit obě současně, mohou i vracet různé hodnoty, nejsou spolu v tomto smyslu nijak provázany.



Obrázek 3.10 Algoritmus funkce SelfTest

3.7 SIMULOVANÝ HARDWARE MULTIMETRU

Tato část programu zpracovává odchycenou VISA komunikaci od IVI ovladače a odpovídá na ni. Simulovaný multimetr si ukládá vlastní parametry nezávisle na IVI atributech, díky tomu je například možné odhalit stav, kdy IVI Engine a samotný přístroj mají odlišné atibuty.

Zachycené VISA funkce jsou :

viOpen - inicializační funkce, volá se na začátku komunikace každé session (z funkce IviInit).

viPrintf - funkce vysílání dat z IVI ovladače na přístroj. Program vyhodnotí přijatý příkaz a přeloží jej na číselný kód, podle kterého může rozhodnout o odpovědi, bude-li vzápětí vyslán dotaz na výsledek. Je-li přijat příkaz na změnu hodnoty atributu, je tento atribut ihned přepsán do proměnných simulovaného přístroje (visa_custom.c).

viScanf - funkce, která očekává odpověď od přístroje. Očekává se, že při správné činnosti programu je každá takováto funkce předcházena voláním funkce viPrintf, která sdělí přístroji, které informace požaduje. Je-li poslední přijatá funkce viPrintf s neznámým obsahem, nebo je-li funkce viScanf volána dřív než viPrintf, je vrácena hodnota "0" typu 32-bitový integer.

viQueryf - je funkce, která kombinuje funkce viPrintf a viScanf, tedy vyšle data do přístroje a zároveň čeká na odpověď. Není-li dotaz pro přístroj srozumitelný, není vyslána žádna odpověď a IVI ovladač přeruší čekání po uplynutí doby timeout (příloha B.4).

viRead a viWrite - podobné jako viPrintf a viScanf, tyto funkce ve své originální podobě neprovádí žadnou konverzi a formátování dat (převod číselných proměnných na znaky), simulovaný hardware je pomocí funkce viRead schopen odpovědět na pokyn k identifikaci přístroje "**IDN*". Funkci viWrite pak pouze vypisuje do záznamového okna (příloha B.3).

Zachycení volaných funkcí je provedeno pomocí makra **#define** (u funkcí s proměnným počtem parametrů je použito variadické makro). Tato zachycení jsou deklarována v souboru visa_custom.h. Např. řádek kódu :

#define viScanf(vi,readFmt, ...) viCustomScanf(vi,readFmt, __VA_ARGS__)

způsobí, že volání funkce viScanf s proměnným počtem parametrů bude při kompilaci přeloženo na volání funkce viCustomScanf, rovněž je zde vidět použitá syntaxe variadického makra. Definice funkce potom vypadá následovně :

```
ViStatus _VI_FUNCC  viCustomScanf(ViSession vi, ViString readFmt, ...)
{
        va_list list;
        ViInt32         *promenna;
        va_start( list, readFmt );
        e = va_arg( list, ViInt32* );
        *promenna = 0;
        va_end( list );
        ...
```

Na tomto zjednodušeném zápisu je ukázáno jak funkce získává hodnoty proměnných. U tohoto způsobu předávání proměnných je nutné, aby vždy byla předána alespoň jedna (mimo neformátovaný text předaný ve ViString readFmt). Proto všechny takto zachycené funkce byly upraveny v multimetr.c ovladači, aby pokud posílají pouze text bez dalších proměnných, poslaly proměnnou integer hodnoty 0. Např. :

```
viPrintf (io, ":RANG:AUTO?",0)
```

Při vytváření programu se ukázalo, že zachytávání komunikace v projektu je dostačující, není třeba tyto funkce nahrazovat umělou VISA dll knihovnou.

4. MODIFIKACE ATRIBUTŮ

Definice atributu - v IVI specifikacích tento termín obecně označuje C atribut nebo COM (Component Object Model) vlastnost. Atributy jsou používány pro konfiguraci hardwaru a obsluhu softwaru. Obecně platí, že každé nastavení hardwaru je asociováno s atributem konfigurace hardwaru. Tyto atributy umožňují uživateli nastavit a získat hodnoty asociovaných přístrojových nastavení. Atributy obsluhy softwaru spíše řídí jak ovladač přístroje pracuje, než že by představovaly konkrétní nastavení přístroje.[4]

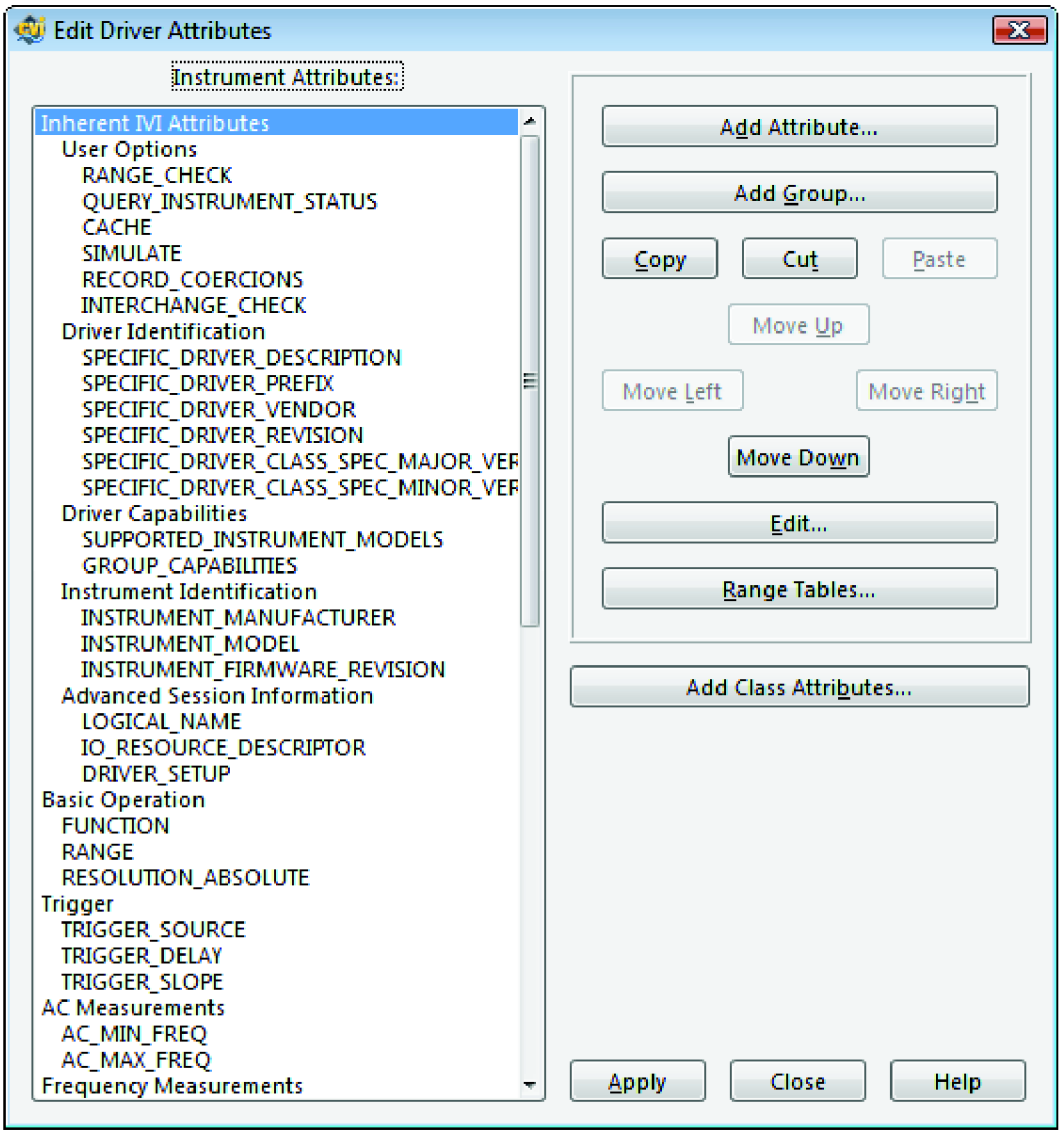
Atributy ovladače můžeme modifikovat přímo přepsáním zdrojového kódu, nebo použít například editor IVI atributů z LabWindows/CVI. Zde bude rozebrán druhý způsob pro obecnou editaci jakéhokoliv IVI ovladače. Zároveň bude popsána potřebná úprava kódu tohoto projektu, která není v editoru zahrnuta. Editor atributů se spouští z nabídky **Tools** – **Edit IVI Specific driver attributes**. Nabídka je přístupná jen tehdy, je-li otevřen soubor ovladače. V editoru pak máme několik možností:

- Add Atribute... Otevře se okno vlastnosti atributu, který chceme přidat.
- Add Group... Otevře okno vlastností skupiny atributů, kterou chceme přidat.
- Copy Kopíruje vybraný atribut nebo skupinu.
- Cut Vyjme vybraný atribut nebo skupinu.
- Paste Vloží naposledy kopírovaný atribut nebo skupinu.
- Move Up Posouvá atribut nebo skupinu v seznamu směrem nahoru.
- Move Down Posouvá atribut nebo skupinu v seznamu směrem dolů.
- Move Left Posouvá atribut nebo skupinu v uspořádání skupin směrem doleva.

[4] URL: <http://www.ivifoundation.org/downloads/Architecture%20Specifications/IVIGlossary-2008-11-17.pdf> [cit. 2009-12-14] přeloženo autorem

- Move Right — Posouvá atribut nebo skupinu v uspořádání skupin směrem doprava.
- Edit — Otevře okno vlastností atributu nebo skupiny, který chceme upravit.
- Range Tables... — Otevře okno úprav tabulek rozsahů.
- Add Class Attributes — Otevře okno přidání třídových atributů (užitečné pokud byl některý například vymazán, protože nemohou být přidány jiné třídové atributy než definované specifikací IVI).
- Apply — Aplikuje provedené změny do zdrojových souborů.
- Close — Zavře okno editoru atributů.
- Help — Otevře nápovědu pro editor atributů.

Hodnoty atributů za běhu programu přímo měnit nelze, je to možné pouze voláním vybraných funkcí. Callback funkce pro zápis atributů a jejich zpracování v simulovaném hardwaru jsou však vypracovány, program je tedy lehce rozšířitelný v tomto směru.

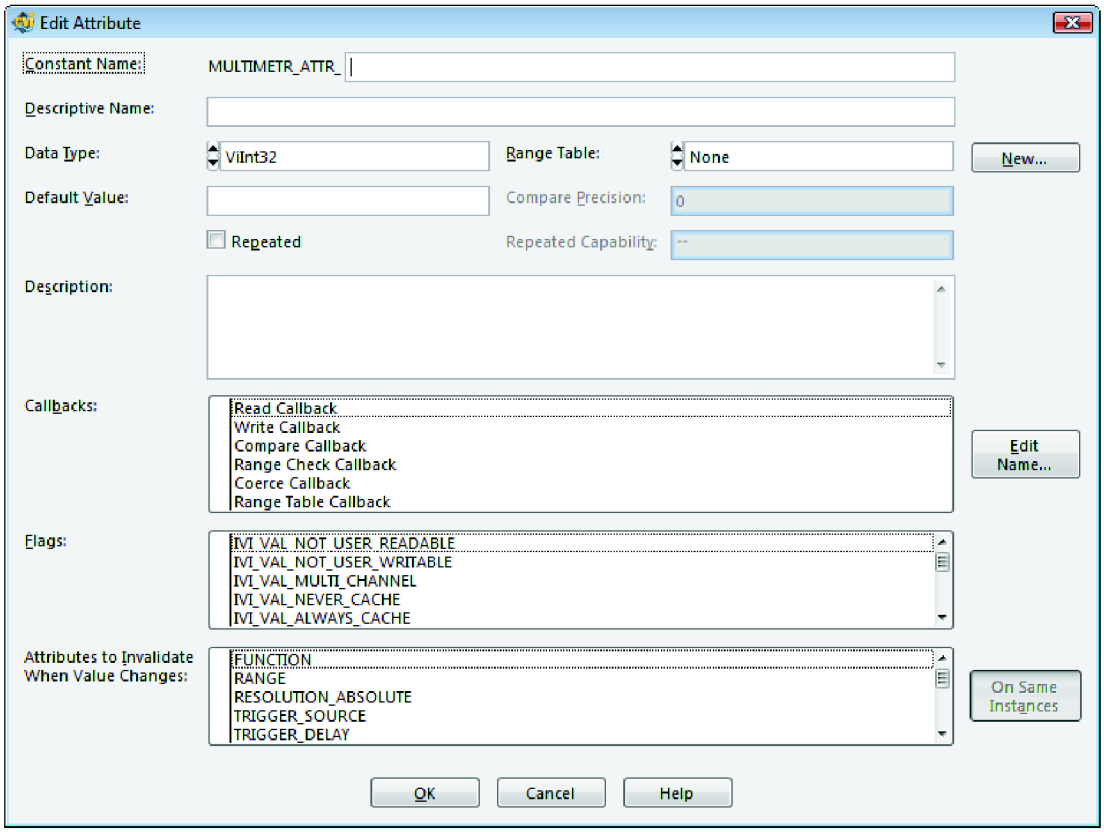


Obrázek 4.1 Okno editoru atributů

4.1 PŘIDÁNÍ ATRIBUTU

Prvky okna jsou stejné jako při editaci atributu a jsou zde popsány. **Constant Name** je název konstanty atributu, předpona „NÁZEV OVLADAČE_ATTR_“ je pevně předdefinována a nelze ji odstranit. **Descriptive Name** je název samotného atributu. **Data Type** je druh proměnné, může být Boolean, Integer, Real, Vi Session, String nebo Vi Addr (typ proměnné není předem daný, ale závisí na konkrétním kontextu). **Range Table** je tabulka rozsahů pro daný atribut. **Default Value** je implicitní

hodnota atributu, tj. hodnota, kterou bude mít atribut po spuštění programu. Pole **Description** slouží ke slovnímu popisu daného atributu. Do seznamu **Callbacks** je možno uvést příslušné Callback funkce atributu (tlačítko **Edit Name** je ke zvolení názvu funkce). **Flags** jsou příznakové bity označující specifikované vlastnosti atributu. **Attributes to Invalidate When Value Changes** je seznam ostatních atributů, jejichž stav je třeba načíst z přístroje, když je hodnota upravovaného atributu změněna.



Obrázek 4.2 Okno přidání atributu

Pro plnou integraci atributu do programu této bakalářské práce je pak třeba navíc vykonat následující:

- přidat zobrazení atributu v seznamu "IVI Attributes" na hlavním panelu při výběru jeho typu v seznamu "Attribute type" do funkce "Ring5CB" (callback funkce seznamu) v souboru bak.c.

- Přidat volání IVI funkce získání hodnoty atributu do callback funkce "ReadAttributeCB" (funkce tlačítka Read Attribute).

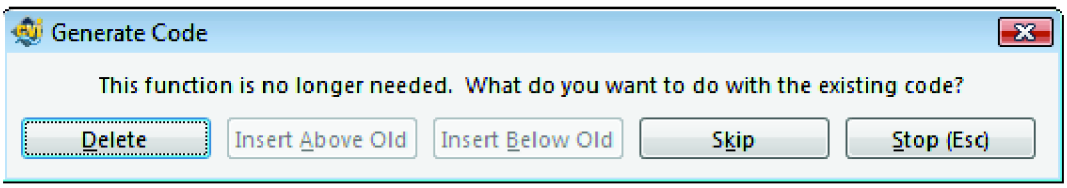
- Zadefinovat atribut v simulovaném hardwaru (soubor visa_custom.c). Jeho typ nemusí nutně odpovídat IVI atributu, například je výhodné ukládat určitý stav, který nabývá několika málo hodnot, jako číselnou proměnnou namísto textové. Při komunikaci s VISA knihovnou je nutná konverze tohoto odlišně definovaného atributu.

- Upravit příslušné VISA funkce v simulovaném hardwaru (visa_custom.c : viRead, viWrite, viPrintf, viScanf). Textový řetězec, kterým je ukládána nebo vyžadována hodnota atributu z přístroje je nadefinován v souboru IVI ovladače (multimetr.c, read a write callback atributu). V případě funkce viPrintf je doporučeno definovat novou "LastPrintCode" hodnotu proměnné pro nový atribut, podle které potom například funkce viScanf pozná, co má odeslat. Pokud je do přístroje zapisována hodnota atributu, pak je nutno ve funkci viPrintf přidat novou 'if' podmínku pro odeslaný příkaz (příkaz je v nenaformátovaném tvaru, tedy např. ":RES %le", kdy hodnota parametru je předávána v jiné proměnné). Po vzoru ostatních atributů je pak možné provést výpis proběhlé komunikace do textboxu na hlavním panelu. Do funkce viScanf je nakonec nutné přidat podmínku 'if' k dané hodnotě "LastPrintCode" pro vyslání naformátované odpovědi typu string.

- Ke korektní funkci atributu v IVI ovladači (multimetr.c) je třeba přidat jejich podmíněné zobrazení na hlavním panelu (nastaveni záznamu dat v Settings panelu).

4.2 ODSTRANĚNÍ ATRIBUTU

Odstranění některého atributu z editoru atributů se provádí jeho označením, a stiskem tlačítka **Cut**. Pokud jsou s atributem asociovány nějaké funkce, program se zeptá, jak s nimi má naložit.



Obrázek 4.3 Okno mazání nepotřebných funkcí

Vrozené atributy nelze mazat. Je třeba dát pozor při mazání třídových atributů, protože při jejich absenci ovladač ztrácí status třídového ovladače (class compliance).

Při mazání atributů integrovaných v této práci, je pak navíc ještě třeba odstranit všechny úpravy popsané v kap. 4.1 (například opomenutí odstranění atributu ze seznamu na hlavním panelu a jeho následný výběr by způsobil pád aplikace).

5. PROVEDENÉ MĚŘENÍ

National Instruments uvádí pro IVI ovladače řadu výhod [5]. Následující z nich byla vybrána pro svou ověřitelnost k proměření :

- ukládání stavu (State Caching)
- volitelná kontrola rozsahu
- volitelný dotaz na stav přístroje
- simulace přístroje

5.1 UKLÁDÁNÍ STAVU

Měření pro udávanou úsporu komunikace při použití state cachingu.
Postup měření :

1. Na hlavním panelu byl nastaven komunikační port "COM3".
2. V panelu třídy jsou nastaveny parametry "ID Query", "Reset Device", "Simulate Device", "Range Check", " Query Instrument Status" na hodnotu "off", a parametr "Cache" na hodnotu "on".
3. Ostatní parametry jsou ponechány na implicitních hodnotách, to jest Range "10.00", Resolution (absolute) "0.001", Measurement function "DC Voltage", Trigger source "Immediate" a trigger delay "0.00".
4. Z hlavního panelu byly volány funkce v následujícím pořadí :
   - InitWithOptions
   - ConfigureMeasurement
   - ConfigureTrigger
   - Read (volána 10x za sebou)
   - SelfTest
   - Close.
5. Měření bylo opakováno s parametrem "Cache" s hodnotou "off".

Počet přenesených znaků se zapnutým state cachingem je 343 a bez něj 580 (sečtena příchozí i odchozí komunikace, viz tab. 1).

Výpočet úspory přenesených dat v procentech je uveden ve vztahu 5.1:

$$u = 100.(1 - \frac{a}{b}) = 100.(1 - \frac{343}{580}) = 40{,}86\% \qquad [\%, byte, byte]. \qquad (5.1)$$

u ... úspora v procentech [%]

a ... přenesená data se state cachingem [byte]

b ... přenesená data bez state cachingu [byte]

Na zkušebním měření se tedy podařilo dosáhnout 40,86% úspory přenesených dat.

*CLS;*ESE 1;*SRE 32 :TRIG:COUN 1 :SAMP:COUN 1 :ZERO:AUTO OFF; DCV; :RANG 1.000000e+01; :RES 1.000000e-04; :TRIG:SOUR IMM; :TRIG:DEL 1.000000e-07; FETC? 4.764855e+00 FETC? 1.861629e-02 FETC? 2.314524e+00 FETC? 8.085879e+00 FETC? 9.063997e-01 FETC? 9.287698e+00 FETC? 8.579363e+00 FETC? 3.938719e+00 FETC? 6.001770e+00 FETC? 2.268746e+00 *TST? 0	*CLS;*ESE 1;*SRE 32 :TRIG:COUN 1 :SAMP:COUN 1 :ZERO:AUTO OFF; DCV; FUNC? DCV FUNC? DCV :RANG 1.000000e+01; :RANG:AUTO? 0 RANGEQUERY? 1.000000e+01 :RANG:AUTO? 0 RANGEQUERY? 1.000000e+01 :RANG:AUTO? 0 RANGEQUERY? 1.000000e+01 :RES 1.000000e-04; :TRIG:SOUR IMM; :TRIG:DEL 1.000000e- FETC? 4.617145e+00 FUNC? DCV FETC? 2.224189e+00 FUNC? DCV FETC? 3.657643e+00 FUNC? DCV FETC? 8.936430e+00 FUNC? DCV FETC? 5.207373e+00 FUNC? DCV FETC? 8.594317e+00 FUNC? DCV FETC? 2.976775e+00 FUNC? DCV FETC? 2.324290e+00 FUNC? DCV FETC? 9.688406e+00 FUNC? DCV FETC? 6.087527e+00 FUNC? DCV *TST? 0

Tabulka 1. Přenesená data s a bez state cachingu

5.2 VOLITELNÁ KONTROLA ROZSAHŮ

Při měření byl program nastaven podobně jako v předchozím případě, všechny inicializační parametry na hodnotě "off". V průběhu byly volány funkce InitWithOptions a ConfigureMeasurement s parametrem "range" na hodnotě 542.00 (hodnota odlišná od validních rozsahů v tabulce range). Experiment byl opakován se zapnutým parametrem Range Check. V obou případech byl odeslán stejný příkaz pro změnu rozsahu přístroje a to *":RANG 1.000000e+03;"*, tedy odeslaný rozsah byl upraven IVI Enginem. Takto je to popsáno i v Instrument Driver Developers Guide [5], kde se uvádí, že v některých případech IVI Engine opravuje hodnotu nehledě na nastavení kontroly rozsahů. Měření bylo zopakováno pro parametr "absolute resolution" s hodnotou 0.003, který byl v obou případech pozměněn na hodnotu 0.0001. IVI Engine tedy disponuje schopností kontroly a opravy rozsahu, avšak se u zvolených atributů nepodařilo prokázat, zda je tato schopnost skutečně volitelná.

5.3 VOLITELNÝ DOTAZ NA STAV PŘÍSTROJE

Program byl nastaven obdobně jako u prvního testu, bez State Cachingu, "Query Instrument status" na hodnotě "on". Byly volány stejné funkce jako v prvním testu (kap. 5.1, bod 4). Při každém jednom volání funkce byla komunikace zakončena dotazem na stav přístroje "**ESR?*" a automatickou odpovědí (implicitně nastavenou na "0" - bez chyby). Pro kontrolu bylo měření zopakováno s "Query Instrument status" přepnutým na "off". Během této komunikace pak nebyl zaregistrován ani jeden dotaz na stav přístroje. V panelu settings při nastavení jiné, chybové odpovědi (např. 4[5]), program rozpozná, že nedošlo ke korektnímu ukončení operace, což se projeví na návratové hodnotě funkce IVI ovladače, jež je přeložena a zobrazena v textboxu IVI ENGINE na hlavním panelu. Můžeme tedy říci, že IVI Engine od National Instruments tuto schopnost má.

---

[5] Odpověď symbolizující chybu není libovolná odlišná od nuly, musí po logickém součtu s hodnotou 0x3C odpovídat konstantě IVI_ERROR_INSTR_SPECIFIC, je možno upravit ve funkci multimetr_CheckStatusCallback v souboru multimetr.c.

5.4 SIMULACE PŘÍSTROJE

resourceName : COM3	Value red : 4.593951 Volts
ID Query : no	Value red : 1.329386 Volts
Reset device : no	Value red : 5.323649 Volts
Simulate device : yes	Value red : 5.462508 Volts
Range Check : no	Value red : 7.669301 Volts
Query instrument status : no	Value red : 2.292856 Volts
Cache : no	Value red : 8.865322 Volts
	Value red : 1.734977 Volts
Measurement function : MULTIMETR_VAL_DC_VOLTS	Value red : 0.966216 Volts
Range : 10.000000	Value red : 4.546648 Volts
Resolution : 0.001000	Value red : 9.934690 Volts
	Selftest result :
Trigger source : MULTIMETR_VAL_IMMEDIATE	No error.
Trigger delay : 0.000000	
	Session 1 closed

Tabulka 2. Textový výstup měření v režimu IVI simulace

Pri měření byly všechny inicializační parametry nastaveny na "off", kromě "Simulate device". Sled odeslaných funkcí byl stejný jako v měření State Cachingu. Žádná přístrojová komunikace VISA neproběhla, práce přístroje tedy byla simulována v IVI ovladači. Funkce odpovídaly (tab. 2) stejným způsobem jako "bez" simulace (tedy simulace na vrstvě nižší než IVI ovladač). National Instruments IVI Instrument driver wizardem vygenerovaný ovladač tedy je schopný simulace vybraných funkcí.

5.5 KONTROLA IVI OVLADAČE

IVI ovladač multimetru byl prověřen také nástrojem IVI Specific Driver Test Suite[6]. Tento nástroj je určen ke kontrole a optimalizaci ovladače vygenerovaného IVI Instrument driver wizardem a obsahuje celkem čtyři testy. Tyto testy spolupracují, proto byly v jednom měření zvoleny všechny. Některé postupy testů jsou možné pouze s použitím IVI Enginu od National Instruments, což je tento

případ (tato skutečnost nebude dále uváděna v popisu postupů jednotlivých testů). Výsledky všech testů jsou uvedeny v přílohách. V následujících kapitolách jsou uvedeny popisy jednotlivých testů.

5.5.1 Behavior test

Test chování ovladače provádí následující kontroly:

- Zkontroluje formát verze DLL souboru. Hlavní i vedlejší verze musí být nezáporná celá čísla menší nebo rovna hodnotě 255.
- V spolupráci se Structure testem ověří shodu verze DLL souboru s verzí v hlavičkovém souboru.
- Ověří, zda ovladač odstraní nepoužité přípony po inicializaci.
- Vypíše třídou definované atributy, které ovladač nemá implementovány.
- Vypíše třídou definované funkce, které ovladač nemá implementovány.
- Pro implementované atributy ověří, zda jsou implementovány způsobem definovaným pro danou třídu následovně:
    - Ověří korektnost názvu atributu.
    - Porovná příznakové bity atributů s třídou definovanými příznakovými bity a ohlásí chybějící nebo nadbytečné příznakové bity.
    - Zkontroluje, zda má atribut správný datový typ.
    - Zkontroluje, zda má atribut správny přístup ke čtení a zápisu a základy opakovatelných schopností.
    - Zkontroluje, zda jsou parametry opravovány do správného směru (nahoru či dolů).
- Ověří, zda jsou funkční parametry ignorovány pod podmínkami definovanými ve specifikacích IVI Foundation.

5.5.2 Compliance test

Test shody s posledními specifikacemi IVI Foundation, provádí následující kontroly:

- Vypíše sady rozšíření, se kterými je ovladač kompatibilní.
- Pokud ovladač s některou sadou rozšíření není kompatibilní, potom test vypíše test důvod následujícím způsobem:
    - Vypíše atributy a funkce, které jsou v sadě rozšíření definovány, ale v ovladači chybí.
    - Vypíše proč ovladač nesplňuje podmínku shody, na základě pravidel shody pro třídu.
- Vypíše základní a rozšířené sady schopností, se kterými je ovladač ve shodě, mimo shodu a které nemá implementovány.

5.5.3 Structure test

Test struktury ovladače, který jako vstup vyžaduje zdrojový kód, provádí následující kontroly:

- Ověří formát verze v hlavičkovém souboru. Hlavní i vedlejší verze musí být nezáporná celá čísla menší nebo rovna hodnotě 255.
- Ověří, zda jsou checkErr a viCheckErr makra použita korektně a konzistentně.
- Vypíše funkce, které nepoužívají uzamykání, mají nevyvážené zamykání sessions a nebo nemají odemknutí v jejich chybovém bloku.
- Vrátí varování pro funkce, které mají jediný návratový příkaz a tudíž nepoužívají uzamykání.
- Vypíše tabulky rozsahů, které existují, ale není na ně odkazováno ve zdrojovém kódu.

- Ověří uživatelskou přístupnost čtení a zápisu a vykoná následující činnosti:

    - Vypíše atributy, které nejsou přidány do session pomocí funkce "Ivi_AddAttribute".

    - Vypíše atributy, které jsou označeny jako skryté, ale jsou definovány v hlavičkovém souboru.

    - Vypíše atributy, které nejsou označeny jako skryté, ale jsou definovány ve zdrojovém kódu.

- Ověří atributy pro vstupně výstupní použití v callback funkcích.

- Vypíše atributy, které jsou v hlavičkovém a zdrojovém kódu a jsou označeny následujícím způsobem:

    - Označeny příznakovým bitem IVI_VAL_USE_CALLBACKS_ _FOR_SIMULATION, ale používají ve svých callback funkcích vstupně výstupní volání.

    - Nejsou označeny příznakovým bitem IVI_VAL_USE_ _CALLBACKS_FOR_SIMULATION, ale nepoužívají ve svých callback funkcích vstupně výstupní volání.

- Vypíše callback funkce, které jsou deklarovány, ale nejsou definovány nebo na ně není odkazováno ve zdrojovém kódu.

- Prověří, že všechny funkce vyšších atributů mají tabulku rozsahů, kontrolní callback, callback tabulky rozsahů nebo implementují dynamickou tabulku rozsahů. Ohlásí varování za každý skrytý atribut, atribut typu bool nebo atribut pouze pro čtení.

- Ověří, zda atributy pouze pro čtení nemají callback funkci pro zápis, která uskutečňuje vstupně výstupní komunikaci.

- Prověří, že konfigurační funkce neprovádějí vstupně výstupní komunikaci.

- Ověří, zda callbacky pro atributy, které nejsou kanálově založené, se nepokoušejí získat, nastavit nebo kontrolovat vícekanálové atributy použitím channelName parametru.

- Ověří, zda funkce Ivi_GetAttribute není volána na atribut označený příznakovým bitem IVI_VAL_NOT_USER_READABLE.

- Ověří, zda funkce Ivi_SetAttribute není volána na atribut označený příznakovým bitem IVI_VAL_NOT_USER_WRITABLE.

- Ověří, zda atributy IVI_VAL_NOT_USER_WRITABLE jsou nastaveny pomocí příznakového bitu IVI_VAL_DONT_MARK_AS_SET_BY_ _USER.

5.5.4 Function Panel test

Test funkčního panelu ovladače, který jako vstup vyžaduje zdrojový kód, provádí následující kontroly:

- Ověří, zda formát souboru funkčního panelu je verze CVI 5.5 nebo starší.
- Ověří, zda předpona ovladače odpovídá názvu ".fp" souboru.
- Ověří, zda bloky CHANGE a PREFIX jsou odstraněny z textu nápovědy.
- Ověří, zda každý první ovládací prvek na všech panelech, kromě funkčních panelů "Prefix_init" a "Prefix_initWithOptions", je prvek typu ViSession pojmenovaný "Instruments Handle" nebo "vi".
- Pro datové typy ViBoolean ověří, zda je ovládací prvek binární.
- Ověří, zda nápověda pro každý prvek obsahuje Default Value, Defined Values, nebo sekci Defined Constant Values a Valid Range sekci, zavísející na typu ovládacího prvku.
- Pro ovládací prvky typu Ring (seznamy), ověří, zda páry název/hodnota odpovídají nápovědě.
- Pro ovládací prvky Ring, ověří, zda pořadí párů název/hodnota odpovídá pořadí v nápovědě.
- Pro datové typy ViConstString ověří, zda-li je korespondující datový typ ovládacího prvku funkčního panelu ViChar[] .
- Ověří, zda prvky značené Channel Name maji sekci Valid Channel Names v nápovědě.
- Ověří, zda velikost a pozice všech funkčních panelů je standardní.

6. ZÁVĚR

Podle zadání byl zhotoven demonstrační program v Labwindows/CVI verze 9.0. Hlavním cílem bylo vytvořit takový program, kterým jdou ověřit udáváné výhody IVI ovladačů. V průběhu práce se ukázalo, že je k tomu potřeba zobrazovat komunikaci už na úrovni komunikace s hardwarem, nikoliv v simulačním režimu IVI ovladače. Toho bylo dosaženo zachycením volání VISA funkcí a jejich nahrazením. Program pracuje s IVI ovladačem nově vytvořeným k tomuto účelu, a sice třídy digitální multimetr. Třída dig. multimetru byla vybrána pro svou jednoduchost. Program dokumentuje činnost IVI ovladače, jeho komunikaci s IVI Enginem a simulovanou VISA komunikaci. Postup práce s programem a jeho modifikace je popsána v dokumentaci.

Výhody IVI ovladačů uváděné firmou National Instruments byly otestovány měřením na simulovaněm hardwaru. Při něm byla zjištěna zásadní úspora komunikace díky state cachingu až o 40% dat a automatická oprava některých nastavení podle tabulky atributu. Rovněž se potvrdilo, že je IVI Engine v prostředí LabWindows/CVI schopen reagovat na chybu stavu přístroje uživatelem definovaným způsobem. Vytvořený přístrojový ovladač byl otestován programem IVI Specific Driver Test Suite od National Instruments. Jediné chyby, které tento program odhalil, jsou v nápovědě funkčního panelu ovladače, jenž nebyla upravena, a obsahuje proto vygenerované popisky, které tedy nejsou zaručeně přesné.

Do budoucna je možné rozšířit program o další třídy a funkce ovladačů, případně výpis záznamů do souboru. Zároveň je plánováno doplnit program o automatickou detekci vhodného portu, čímž se zjednoduší jeho obsluha.

ZDROJE INFORMACÍ

[1] IVI Foundation : Getting Started Guide [online]. Revision 1.01,[cit. 2009-11-16], 102 s. Dostupný na : <http://www.ivifoundation.org/downloads/IVI_GSG%20v1.01.pdf>

[2] IVI Foundation : IVI-5.0: Glossary [online]. Revision 1.03, [cit. 2009-12-14], 13 s. Dostupný na: <http://www.ivifoundation.org/downloads/Architecture%20Specifications/IVIGlossary-2008-11-17.pdf>

[3] LabWindows/CVI 9.0 [počítačový program] Ver. 9.0.1(375), [cit. 2009-12-1], Vývojové prostředí od National Instruments

[4] National Instruments : IVI Foundation Overview [online]. [cit. 2009-12-5], 16 s. Dostupný na : <ftp://ftp.ni.com/pub/devzone/IVIFoundation.zip>

[5] LabWindows/CVI Instrument Driver Developers Guide[online]. April 2003 Edition, [cit. 2010-5-1] , 193 s. Dostupný na : <http://www.ni.com/pdf/manuals/370699a.pdf >

[6] National Instruments: IVI Specific Driver Test Suite[počítačový program] Ver. 4.1.0f2 [cit. 2010-1-5]. Počítačový program pro testaci IVI ovladačů.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

COM	Sériový komunikační port
VISA	Virtual Instrument Software Architecture, Softwarová architektura virtuálních nástrojů
IVI	Interchangeable Virtual Instruments, zaměnitelné virtuální nástroje
SCPI	Standard Commands for Programmable Instrumentation, standardní příkazy pro progamovatelné nástroje
GPIB	General Purpose Interface Bus, rozhraní sběrnice pro obecné použití
DLL	Dynamic link library, dynamicky propojená knihovna
CVI	C for Virtual Instrumentation, jazyk C pro virtuáĺní nástroje

SEZNAM PŘÍLOH

A NÁVOD K OBSLUZE.....45
A.1 SPUŠTĚNÍ PROGRAMU.....45
A.2 NASTAVENÍ PROGRAMU.....45
A.3 PRÁCE S PROGRAMEM A JEJÍ UKONČENÍ.....45
B ZDROJOVÝ KÓD.....46
B.1 UKÁZKA FUNKCE MeasureProceed.....46
B.2 FUNKCE ConfigureTrigger IVI OVLADAČE.....47
B.3 FUNKCE viCustomRead SIMULOVANÉHO HARDWARU.....47
B.4 FUNKCE viCustomQueryf SIMULOVANÉHO HARDWARU..... 48
C - VÝSLEDKY TESTŮ "IVI SPECIFIC DRIVER TEST SUITE".....54

PŘÍLOHA A - NÁVOD K OBSLUZE

A.1 SPUŠTĚNÍ PROGRAMU

Program se spouští pomocí dodaného .exe souboru. Jeho funkce je ověřena na operačním systému Microsoft Windows 7 Professional 64bit. K jeho správnemu chodu je třeba mit nainstalován LabWindows/CVI verze 9.0 nebo vyšší a ovladače VISA.

A.2 NASTAVENÍ PROGRAMU

Po spuštění je nutno zkontrolovat, zda-li je zadán existující komunikační port v textovém poli "Device connection". Porty je možno zkontrolovat pomocí programu "Measurement & Automation explorer" od National Instruments a sice v záložce My System - Devices and Interfaces - Serial & Parallel (program by měl fungovat i v jiných VISA sběrnicích, např. GPIB, ale otestován byl pouze na sériovém a paralelním portu). Při špatném zadání či zvolení nevhodného portu program při pokusu o vytvoření přístrojové session na toto upozorní uživatele chybovou zprávou a nevytvoří session. Dále je vhodné nastavit zaznamenávaná data pomocí tlačítka Settings.

A.3 PRÁCE S PROGRAMEM A JEJÍ UKONČENÍ

Práce spočívá ve výběru požadované funkce na hlavním panelu a její potvrzení tlačítkem Perform. Jakákoliv nastavení provedená v třídovém panelu se berou v potaz vždy až po vykonání specifické funkce (tedy např. zvolení rozsahu přístroje se provede až po odeslání funkce ConfigureMeasurement). Jednotlivé atributy přístroje lze v kterémkoliv okamžiku číst tlačítkem Read Attribute (předpoklad je aktivní alespoň jedna session). Při ukončení práce je doporučeno uzavřít jednotlivé sessions. Pokud nejsou ukončeny, program se je pokusí ukončit automaticky. Manuální ukončování je vhodné zejména při práci s více sessions najednou v případě, že je potřeba pokračovat v práci pouze s některými. Úplné ukončení programu se provádí tlačítkem Quit.

PŘÍLOHA B - ZDROJOVÝ KÓD

B.1 UKÁZKA FUNKCE MeasureProceed

```
case 3:
        if(Session[ActiveSession].exist == 0) break;
        ViInt32 triggersource;
        GetCtrlVal (dmmpanel, DMMPANEL_RING_2, &temp);
        switch(temp)
        {
                case 0:
                        triggersource = MULTIMETR_VAL_IMMEDIATE;
                        sprintf(tempstr,"Trigger source : MULTIMETR_VAL_IMMEDIATE");
                        break;
                case 1:
                        triggersource = MULTIMETR_VAL_EXTERNAL;
                        sprintf(tempstr,"Trigger source : MULTIMETR_VAL_EXTERNAL");
                        break;
                case 2:
                        triggersource = MULTIMETR_VAL_SOFTWARE_TRIG;
                        sprintf(tempstr,"Trigger source : MULTIMETR_VAL_SOFTWARE_TRIG");
                        break;
                        default :

                        break;
        };

        double triggerdelay;
        GetCtrlVal (dmmpanel, DMMPANEL_NUMERIC_3, &triggerdelay);

        status = multimetr_ConfigureTrigger (Session[ActiveSession].session, triggersource, triggerdelay);
        InsertTextBoxLine (testpanel, TESTPANEL_TEXTBOX , -1, "multimetr_ConfigureTrigger");
        InsertTextBoxLine (testpanel, TESTPANEL_TEXTBOX , -1, "Operation result :");
        multimetr_error_message (Session[ActiveSession].session, status, errString);
        InsertTextBoxLine (testpanel, TESTPANEL_TEXTBOX , -1, errString);
        InsertTextBoxLine (testpanel, TESTPANEL_TEXTBOX , -1, "");
        InsertTextBoxLine (testpanel, TESTPANEL_TEXTBOX2 , -1, tempstr);
        sprintf(tempstr,"Trigger delay : %f",triggerdelay);
        InsertTextBoxLine (testpanel, TESTPANEL_TEXTBOX2 , -1, tempstr);
        InsertTextBoxLine (testpanel, TESTPANEL_TEXTBOX2 , -1, "");
        break;
```

B.2 FUNKCE ConfigureTrigger IVI OVLADAČE

```
/*****************************************************************************
 * Function: multimetr_ConfigureTrigger
 * Purpose:  Configures the common DMM trigger attributes.  These attributes
 *           are MULTIMETR_ATTR_TRIGGER_SOURCE and
 *           MULTIMETR_ATTR_TRIGGER_DELAY.
 *****************************************************************************/
ViStatus _VI_FUNC multimetr_ConfigureTrigger (ViSession vi, ViInt32 triggerSource,
                                              ViReal64 triggerDelay)
{
    ViStatus    error = VI_SUCCESS;

    checkErr( Ivi_LockSession (vi, VI_NULL));
            if(Log1)InsertTextBoxLine (testpanel, TESTPANEL_TEXTBOX4 , -1, "locked session");

        /* Set attributes: */
    viCheckParm( Ivi_SetAttributeViInt32 (vi, VI_NULL, MULTIMETR_ATTR_TRIGGER_SOURCE,
                                          0, triggerSource), 2, "Trigger Source");
            sprintf(tempstring, "writes attribute : MULTIMETR_ATTR_TRIGGER_SOURCE to %i",triggerSource);
            if(Log4)InsertTextBoxLine (testpanel, TESTPANEL_TEXTBOX4 , -1, tempstring);
    viCheckParm( Ivi_SetAttributeViReal64 (vi, VI_NULL, MULTIMETR_ATTR_TRIGGER_DELAY,
                                           0, triggerDelay), 3, "Trigger Delay");
            sprintf(tempstring, "writes attribute : MULTIMETR_ATTR_TRIGGER_DELAY to %f",triggerDelay);
            if(Log4)InsertTextBoxLine (testpanel, TESTPANEL_TEXTBOX4 , -1, tempstring);

    checkErr( multimetr_CheckStatus (vi));

Error:
    Ivi_UnlockSession(vi, VI_NULL);
            if(Log1)InsertTextBoxLine (testpanel, TESTPANEL_TEXTBOX4 , -1, "unlocked session");
    return error;
}
```

B.3 FUNKCE viCustomRead SIMULOVANÉHO HARDWARU

```
ViStatus _VI_FUNC  viCustomRead(ViSession vi, ViPBuf buf, ViUInt32 cnt, ViPUInt32 retCnt)
{
            if(Log2)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, "viRead called");

            if(LastPrintCode == 1)
            {
                        buf[0] = 'M';
                        buf[1] = 'U';
```

```
                buf[2] = 'L';
                buf[3] = 'T';
                buf[4] = 'I';
                buf[5] = 'M';
                buf[6] = 'E';
                buf[7] = 'T';
                buf[8] = 'R';
                *retCnt = 9;
                if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, ">>> MULTIMETR");
        }

        return VI_SUCCESS;
};
```

B.4 FUNKCE viCustomQueryf SIMULOVANÉHO HARDWARU

```
ViStatus _VI_FUNCC viCustomQueryf(ViSession vi, ViString writeFmt, ViString readFmt, ...)
{
        if(Log2)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, "viQuery called");
        if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, writeFmt);
        if(strcmp(writeFmt,":RES?")==0)
        {
                va_list         list;
                ViReal64            *e;
                va_start( list, readFmt );
                e = va_arg( list, ViReal64* );
                *e = absRes;
                va_end( list );
                char tempstr[255];
                sprintf(tempstr,">>> %le", absRes);
                if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
        }

        if(strcmp(writeFmt,":%s:APER:UNITS?")==0)
        {
                va_list         list;
                ViInt32             *e;
                va_start( list, readFmt );
                e = va_arg( list, ViInt32* );
                *e = 1;
                va_end( list );
                char tempstr[255];
                sprintf(tempstr,">>> %le", absRes);
                if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
        }
```

```
else if(strcmp(writeFmt,":ZERO:AUTO?")==0)
{
		va_list		list;
		ViInt32			*count;
		ViChar	*buff;
		va_start( list, readFmt );
		count = va_arg( list, ViInt32* );
		buff =  va_arg( list, ViChar* );

		switch (AutoZero)
		{
			case MULTIMETR_VAL_AUTO_ZERO_OFF:
			buff[0] = 'O';buff[1] = 'F';buff[2] = 'F';buff[3] = '\0';
			*count = 3;
			 break;
			case MULTIMETR_VAL_AUTO_ZERO_ON:
			buff[0] = 'O';buff[1] = 'N';buff[2] = '\0';
			*count = 6;
			break;
			case MULTIMETR_VAL_AUTO_ZERO_ONCE:
			buff[0] = 'O';buff[1] = 'N';buff[2] = 'C';buff[3] = 'E';buff[4] = '\0';
			*count = 4;
			break;
			default:
			break;
		}

		va_end( list );
		char tempstr[255];
		sprintf(tempstr,">>> ");
		strcat(tempstr,&buff[0]);
		if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
}
else if(strcmp(writeFmt,":TRANS:TYPE?")==0)
{

		va_list		list;
		ViInt32			*count;
		ViChar	*buff;
		va_start( list, readFmt );
		count = va_arg( list, ViInt32* );
		buff =  va_arg( list, ViChar* );
```

```
		switch (TransducerType)
		{
			case MULTIMETR_VAL_THERMOCOUPLE:
			buff[0] = 'T';buff[1] = 'C';buff[2] = '\0';
			*count = 3;
			break;
			case MULTIMETR_VAL_THERMISTOR:
			buff[0] = 'T';buff[1] = 'H';buff[2] = 'E';buff[3] = 'R'; buff[4] = 'M';buff[5] = '\0';
			*count = 6;
			break;
			case MULTIMETR_VAL_2_WIRE_RTD:
			 buff[0] = 'R';buff[1] = 'T';buff[2] = 'D';buff[3] = '\0';
			*count = 4;
			break;
			case MULTIMETR_VAL_4_WIRE_RTD:
			buff[0] = 'F';buff[1] = 'R';buff[2] = 'T';buff[3] = 'D';buff[4] = '\0';
			*count = 5;
			break;
			default:
			break;
		}

		va_end( list );
		char tempstr[255];
		sprintf(tempstr,">>> ");
		strcat(tempstr,&buff[0]);
		if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
}
else if(strcmp(writeFmt,":TC:REFJ:TYPE?")==0)
{
		va_list 	list;
		ViInt32 			*count;
		ViChar 	*buff;
		va_start( list, readFmt );
		count = va_arg( list, ViInt32* );
		buff =  va_arg( list, ViChar* );

		switch (ThermocoupleReferenceJunctionType)
		{
			case MULTIMETR_VAL_TEMP_REF_JUNC_INTERNAL:
			buff[0] = 'I';buff[1] = 'N';buff[2] = 'T';buff[3] = '\0';
			*count = 4;
			break;
			case MULTIMETR_VAL_TEMP_REF_JUNC_FIXED:
			buff[0] = 'F';buff[1] = 'I';buff[2] = 'X';buff[3] = '\0';
```

```
                        *count = 4;
                        break;
                        default:
                        break;
                }
                va_end( list );
                char tempstr[255];
                sprintf(tempstr,">>> ");
                strcat(tempstr,&buff[0]);
                if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
}
else if(strcmp(writeFmt,":TC:TYPE?")==0)
{
                va_list         list;
                ViInt32                 *count;
                ViChar     *buff;
                va_start( list, readFmt );
                count = va_arg( list, ViInt32* );
                buff =  va_arg( list, ViChar* );
                buff[0] = '\0';
                *count = 1;
                va_end( list );
                char tempstr[255];
                sprintf(tempstr,">>> ");
                strcat(tempstr,&buff[0]);
                if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
}
else if(strcmp(writeFmt,":FREQ:VOLT:RANGE?")==0)
{
                va_list         list;
                ViReal64                *e;
                va_start( list, readFmt );
                e = va_arg( list, ViReal64* );
                *e = FreqVoltRang;
                va_end( list );
                char tempstr[255];
                sprintf(tempstr,">>> %le",*e);
                if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
}
else if(strcmp(writeFmt,":TC:REFJ:FIXED?")==0)
{
                va_list         list;
                ViReal64                *e;
                va_start( list, readFmt );
                e = va_arg( list, ViReal64* );
```

```
		*e = TempTcFixedRefJunc;
		va_end( list );
		char tempstr[255];
		sprintf(tempstr,">>> %le",*e);
		if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
}
else if(strcmp(writeFmt,":RTD:RES?")==0)
{
		va_list		list;
		ViReal64		*e;
		va_start( list, readFmt );
		e = va_arg( list, ViReal64* );
		*e = RTDRes;
		va_end( list );
		char tempstr[255];
		sprintf(tempstr,">>> %le",*e);
		if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
}
else if(strcmp(writeFmt,":RTD:ALPHA?")==0)
{
		va_list		list;
		ViReal64		*e;
		va_start( list, readFmt );
		e = va_arg( list, ViReal64* );
		*e = RTDAlpha;
		va_end( list );
		char tempstr[255];
		sprintf(tempstr,">>> %le",*e);
		if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
}
else if(strcmp(writeFmt,":THERM:RES?")==0)
{
		va_list		list;
		ViReal64		*e;
		va_start( list, readFmt );
		e = va_arg( list, ViReal64* );
		*e = ThermRes;
		va_end( list );
		char tempstr[255];
		sprintf(tempstr,">>> %le",*e);
		if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
}
else if(strcmp(writeFmt,":%s:APER?")==0)
{
		va_list		list;
		ViReal64		*e;
```

```
                va_start( list, readFmt );
                e = va_arg( list, ViReal64* );
                *e = AperTime;
                va_end( list );
                char tempstr[255];
                sprintf(tempstr,">>> %le",*e);
                if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
        }
        else if(strcmp(writeFmt,":PWLN;FREQ?")==0)
        {
                va_list         list;
                ViReal64                *e;
                va_start( list, readFmt );
                e = va_arg( list, ViReal64* );
                *e = PwlnFreq;
                va_end( list );
                char tempstr[255];
                sprintf(tempstr,">>> %le",*e);
                if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
        }
        else if(strcmp(writeFmt,"*ESR?")==0)
        {
                va_list         list;
                ViInt16                 *e;
                va_start( list, readFmt );
                e = va_arg( list, ViInt16* );
                *e = 0;
                *e = ESRreturn;
                va_end( list );
                char tempstr[255];
                sprintf(tempstr,">>> %ld",*e);
                if(Log3)InsertTextBoxLine(testpanel, TESTPANEL_TEXTBOX4 , -1, tempstr);
        };


        return VI_SUCCESS;
};
```

PŘÍLOHA C - VÝSLEDKY TESTŮ "IVI SPECIFIC DRIVER TEST SUITE"

Behavior Test

Class: ***DMM***
Specific Instrument: **multimetr**
Module Path: **c:\skola\bakalarka\bakalarka.dll**
Test Date: **05-10-2010, 09:43:55**

The DLL version format is valid.
All attributes have correct names.
All attributes have correct data types.
All attributes have correct flags.
All attributes are coerced in the correct direction.

Compliance Test

Class: ***DMM***
Specific Instrument: **multimetr**
Module Path: **c:\skola\bakalarka\bakalarka.dll**
Test Date: **05-10-2010, 09:43:58**

The driver is compliant with the *Inherent* group.
The driver is compliant with the *IviDmmFrequencyMeasurement* group.
The driver is compliant with the *IviDmmACMeasurement* group.
The driver is compliant with the *IviDmmPowerLineFrequency* group.
The driver is compliant with the *IviDmmAutoZero* group.
The driver is compliant with the *IviDmmAutoRangeValue* group.
The driver is compliant with the *IviDmmTriggerSlope* group.
The driver is compliant with the *IviDmmDeviceInfo* group.
The driver is compliant with the *IviDmmSoftwareTrigger* group.
The driver is compliant with the *IviDmmThermistor* group.
The driver is compliant with the *IviDmmResistanceTemperatureDevice* group.
The driver is compliant with the *IviDmmThermocouple* group.
The driver is compliant with the *IviDmmTemperatureMeasurement* group.
The driver is compliant with the *IviDmmMultiPoint* group.
The driver is compliant with the *IviDmmBase capability* group.

Compliance Test Summary:

The driver is compliant with the *Inherent* capability group.

The driver is compliant with the following *IviDmm* groups:

IviDmmBase
IviDmmACMeasurement
IviDmmFrequencyMeasurement
IviDmmTemperatureMeasurement
IviDmmThermocouple
IviDmmResistanceTemperatureDevice

IviDmmThermistor
IviDmmMultiPoint
IviDmmTriggerSlope
IviDmmSoftwareTrigger
IviDmmDeviceInfo
IviDmmAutoRangeValue
IviDmmAutoZero
IviDmmPowerLineFrequency

Structure Test

Specific Instrument: **multimetr**
Source Code File Path: **c:\skola\bakalarka\multimetr\multimetr.c**
Test Date: **05-10-2010, 09:43:59**

WARNING: The following Boolean, hidden, or read-only attributes do not implement range checking by any method:

MULTIMETR_ATTR_APERTURE_TIME
MULTIMETR_ATTR_APERTURE_TIME_UNITS
MULTIMETR_ATTR_AUTO_RANGE_VALUE
MULTIMETR_ATTR_ID_QUERY_RESPONSE
MULTIMETR_ATTR_OPC_TIMEOUT
MULTIMETR_ATTR_VISA_RM_SESSION
MULTIMETR_ATTR_OPC_CALLBACK
MULTIMETR_ATTR_CHECK_STATUS_CALLBACK
MULTIMETR_ATTR_USER_INTERCHANGEx_CHECK_CALLBACK

WARNING: The following functions have a single return statement and do not use locking:

multimetr_ConfigureMeasCompleteDest
multimetr_ConfigureFrequencyVoltageRange
multimetr_ConfigureTransducerType
multimetr_ConfigureFixedRefJunction
multimetr_ConfigureThermistor
multimetr_GetAutoRangeValue
multimetr_ConfigureTriggerSlope
multimetr_ConfigureAutoZeroMode
multimetr_ConfigurePowerLineFrequency
multimetr_ClearError
multimetr_GetNextInterchangeWarning
multimetr_ResetInterchangeCheck
multimetr_ClearInterchangeWarnings
multimetr_InvalidateAllAttributes

FP File Test

Specific Instrument: **multimetr**
Source Code File Path: **c:\skola\bakalarka\multimetr\multimetr.c**
FP File Path: **c:\skola\bakalarka\multimetr\multimetr.fp**
Test Date: **05-10-2010, 09:43:59**

ERROR: The following nodes contain a CHANGE block in the help:

Specific Measurements
Measurement Operation Options
Configuration Information

ERROR: In the *multimetr_init* function panel:

The *Resource Name* control contains a CHANGE block in the help.

The *Status* control contains a CHANGE block in the help.

.....

ERROR: In the *multimetr_ReadInstrData* function panel:

The *Status* control contains a CHANGE block in the help.

WARNING: In the *ReadInstrData* function panel:

The default value for the *Number of Bytes To Read* control does not match the default value in the help. Make sure there is at least one white space between the value and its unit:
Control Default Value: 256
Help Default Value: 0

Výsledek posledního testu není úplný, v místě označeném "....." jsou vynechány chyby typu "... contains a CHANGE block in the help", které jsou v každé funkci, neboť nápověda funkčního panelu nebyla upravována.