Úloha 11 Fyzikální praktikum 1
Programování měřicí karty ADVANTECH PCI-1710HG

1. Měřicí karta ADVANTECH

Vícebodové měření teploty je v úloze realizováno multifunkční měřicí kartou PCI-1710HG firmy ADVANTECH. Karta 1710HG je rozšiřující počítačová karta pro sběrnici PCI. Obsahuje obvody pro analogově-digitální převod, digitálně-analogový převod, digitální vstupy a výstupy a čítač. Analogově digitální převod (též A/D konverze) je proces interpretace měřené analogové veličiny (konkrétně elektrického napětí) celým číslem, které je v počítači dále zpracováváno. Naproti tomu digitálně-analogový převod je opačný postup, při kterém je k zadanému číslu v počítači generována odpovídající hodnota analogové veličiny. Analogově-digitální převod tak počítač staví do role měřicího přístroje, digitálně-analogový převod do role regulovatelného zdroje (malého výkonu) či do role generátoru různých napěťových signálů.

Obrázek 1: Karta PCI-1710HG ve spojení s konektorovým blokem.

rozlišení	**12 bitů**
počet analogových vstupních kanálů	**16** jednostranných nebo 8 rozdílových
rozsah vstupního napětí	**max. 10 V**, viz tab. 2
vstupní impedance	1 GΩ
doba konverze	8 μs
paměť FIFO	4K vzorků
počet analogových výstupních kanálů	**2**
rozsah výstupního napětí	0 – 5 V, **0 – 10 V**
počet digitálních vstupních kanálů	16
počet digitálních výstupních kanálů	16

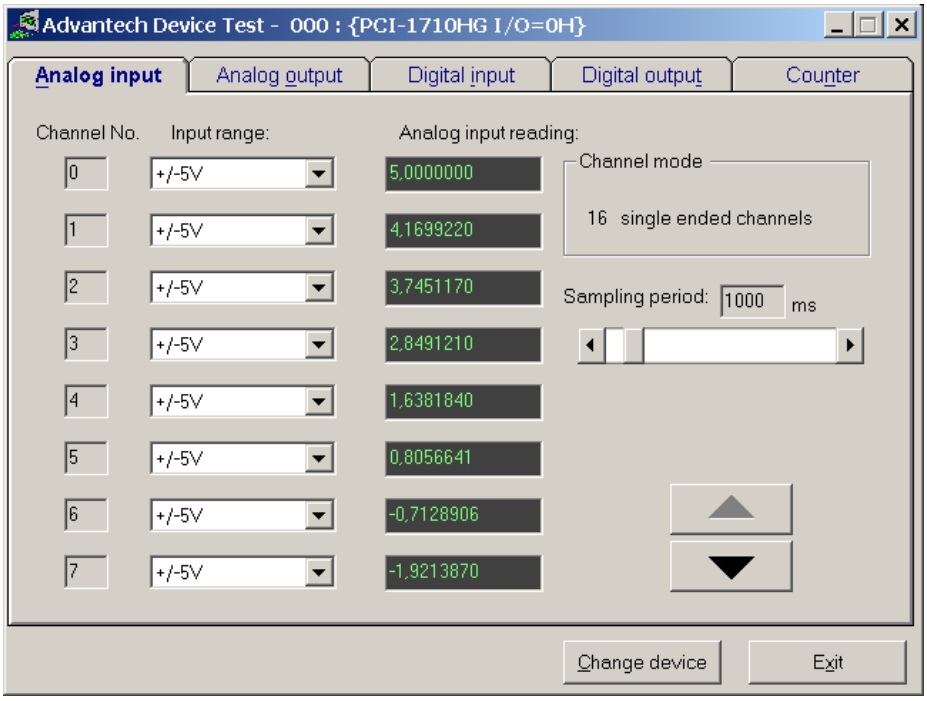
Tabulka 1: Parametry měřicí karty PCI-1710HG

Parametry měřicí karty (vzorkovací frekvence snímání analogového signálu, počet bitů převedeného čísla, vstupní a výstupní rozsah napětí atd.) podstatně ovlivňují možnosti použití počítače pro měření. Parametry karty PCI-1710HG jsou shrnuty v tabulce 1.
Multifunkčnost karty PCI-1710HG spočívá nejen v přítomnosti všech výše uvedených obvodů pro analýzu a generování signálů, ale i v širokém výběru rozsahů měřeného napětí, který umožňuje přesné měření napěťových signálů různé úrovně (viz tabulka 2).

Rozsah	Vzorkovací frekvence (vz./s)	Chyba měření
Bipolární		
± 10 V	100 K	0,01 % of FSR ± 1 LSB
± 5V	100 K	0,01 % of FSR ± 1 LSB
± 1V	35 K	0,02 % of FSR ± 1 LSB
± 0,5V	35 K	0,02 % of FSR ± 1 LSB
± 0,1V	7 K	0,04 % of FSR ± 1 LSB*
± 0,05V	7K	0,04 % of FSR ± 1 LSB*
± 0,01V	770	0,08 % of FSR ± 1 LSB*
± 0,005V	770	0,08 % of FSR ± 1 LSB*
Unipolární		
0 – 10V	100 K	0,01 % of FSR ± 1 LSB
0 – 1V	35 K	0,02 % of FSR ± 1 LSB
0 – 0,1V	7 K	0,04 % of FSR ± 1 LSB*
0 – 0,01V	770	0,08 % of FSR ± 1 LSB*

* Je-li vstup zapojen jako rozdílový.

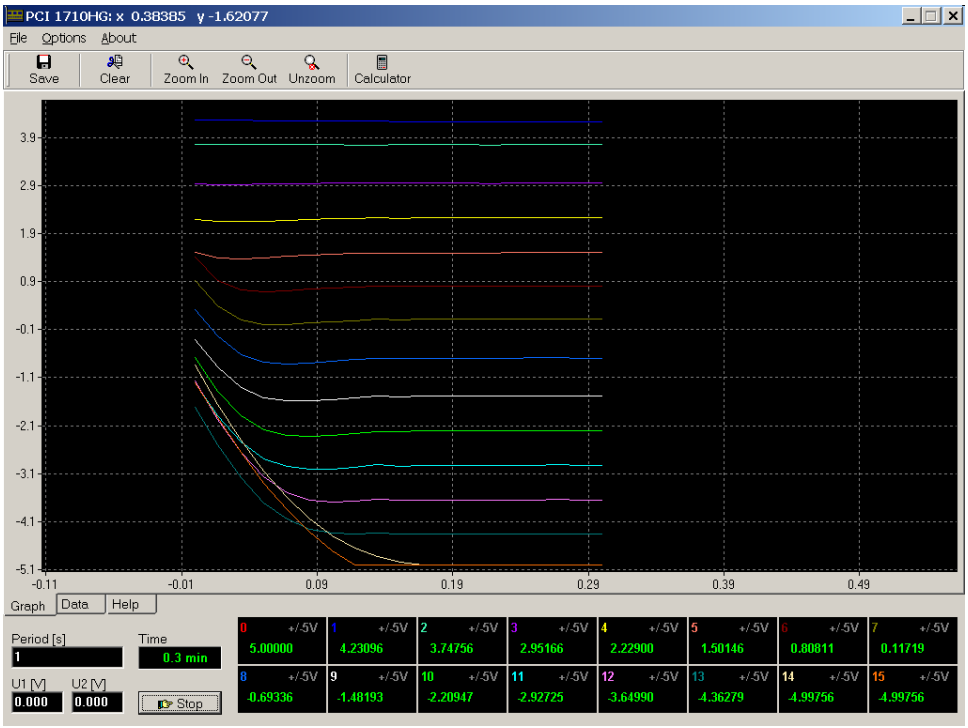
Tabulka 2: Měřicí rozsahy karty PCI-1710HG.



Obrázek 2: Napětí na vstupních kanálech zobrazuje i okno ovladače měřicí karty.

2. Ovládací program

Jednoduché odečtení kartou právě měřených hodnot umožňuje již ovladač měřicí karty (viz. obr. 2). Pro pohodlné měření byl v prostředí Borland Delphi 5.0 vyvinut program PCI-1710HG (obr. 3), který poskytuje možnost zobrazování a ukládání naměřených dat.



Obrázek 3: Program pro měření s kartou ADVANTECH PCI-1710HG.

V programu lze snadno nastavovat

- interval mezi měřeními
- počet aktivních kanálů
- rozsah měření napětí jednotlivých kanálů
- napětí obou výstupních kanálů

Nastavení intervalu

Jednotlivá měření jsou spouštěna softwarově pomocí systémového objektu *timer*. Interval mezi měřeními je tedy zdola omezen hodnotou, která vychází z časového kvanta přidělovaného ve víceúlohovém operačním systému jednotlivým procesům. V programu lze periodu nastavit od 0,1 s výše. Všechny kanály jsou přitom měřeny prakticky současně. Periodu lze měnit i v průběhu měření.

Volba a počet kanálů

Počet kanálů je roven 16 (8) podle nastavení ovladače karty. 16 kanálů je k dispozici v případě, že všechny analogové vstupy jsou používány jako jednostranné (*single ended)*. Kanály je nutné vybírat v souvislém intervalu, což je dáno používanými funkcemi knihovny DLL. Rozsah používaných kanálů lze změnit kliknutím pravého (levého) tlačítka myši nad okénkem kanálu, čímž dojde k nastavení horní (dolní) meze intervalu použitých kanálů. Aktivní kanál je označen symbolem *'ON'*, neaktivní symbolem *'OFF'*.

Volba rozsahu kanálů

U každého kanálu je možné nastavit rozsah měřeného napětí (tzv. *gain*). Ten určuje, v jakém intervalu napětí bude naměřené dvanáctibitové číslo interpretováno, a umožňuje tak podstatně zvýšit rozlišení karty při měření malých napětí. Změna rozsahu se u aktivního kanálu provede jednoduše výběrem ze seznamu nabízených hodnot, který odpovídá přehledu rozsahů uvedeném v tabulce 2.

Napětí výstupních kanálů
Napětí lze volit v rozsahu 0 – 10 V při správně nastaveném ovladači. Hodnoty napětí zapsané do okének jsou okamžitě předány ovladači karty. Desetinným oddělovačem je tečka. Hodnoty lze měnit v průběhu měření.

Měřené hodnoty jsou zobrazovány
- v okénkách příslušných kanálů
- v grafu
- v komponentě *Memo,* desetinným oddělovačem je tečka nebo čárka podle nastavení v menu. Obsah lze uložit do textového souboru příkazem *File|Save as* v menu.

Ovládání grafu
- **zvětšení části grafu (výřez)** se provede výběrem oblasti tažením myši při stisklém levém tlačítku
- **zrušení zvětšení** – dvojitým stiskem levého tlačítka myši
- **posun výřezu grafu** – tažením myši při stisklém pravém tlačítku

3. Programování karty

Program PCI-1710HG dovoluje pouze jednoduché měření napětí a nastavení výstupních kanálů. V případě, že po počítači požadujeme řízení experimentu (např. udržování konstantní teploty), musí počítač umět na hodnoty svých vstupů reagovat nastavováním svých výstupů. Protože tento požadavek nelze v univerzálním programu zabezpečit, je nutné, aby si uživatel napsal potřebný program sám.

```
Function DRV_DeviceGetSubList(DeviceNum : Longint;
Var SubDevList : PT_DEVLIST; MaxEntries : Smallint;
Var OutEntries : Smallint) : Longint; stdcall;
Function DRV_SelectDevice( hCaller : HWND; wGetModule : Boolean;
Var lDeviceNum : Longint;  szDescription : Pchar) : Longint; stdcall;
Function DRV_DeviceOpen(DeviceNum : Longint;
Var DriverHandle : Longint) : Longint; stdcall;
Function DRV_DeviceClose(Var DriverHandle : Longint) : Longint; stdcall;
Function DRV_DeviceGetFeatures(DriverHandle : Longint;
Var lpDeviceGetFeatures : PT_DeviceGetFeatures) : Longint; stdcall;
Procedure DRV_GetErrorMessage(lError : Longint; lpszErrMsg : Pointer);
stdcall;
Function DRV_AIConfig(DriverHandle : Longint;
Var lpAIConfig : PT_AIConfig) : Longint; stdcall;
Function DRV_AIGetConfig(DriverHandle : Longint;
Var lpAIGetConfig : PT_AIGetConfig) : Longint; stdcall;
Function DRV_AIVoltageIn(DriverHandle : Longint;
Var lpAIVoltageIn : PT_AIVoltageIn) : Longint; stdcall;
Function DRV_AIVoltageInExp(DriverHandle : Longint;
Var lpAIVoltageInExp : PT_AIVoltageInExp) : Longint; stdcall;
```

Obrázek 4: Část *interface unit* driver.pas pro měřicí zařízení ADVANTECH.

Měřicí kartu je obecně možné programovat na různé úrovni. Nejnižší, ale také nejobtížnější, je ovládání karty přímým zápisem do paměťových míst (registrů) měřicí karty. Tento postup však vyžaduje detailní znalost karty a hardwaru počítače. Mnohem snadnější je využití některé knihovny rutin pro řízení karty. Výrobce obvykle spolu s kartou dodává kromě ovladače karty i knihovnu DLL, která obsahuje procedury a funkce pro ovládání karty. Aby

knihovna DLL mohla být v programu použita, je nutné mít (nebo si vytvořit) jednotku (tzv. *interface unit*) napsanou v příslušném programovacím jazyce, jež obsahuje odkazy na funkce v knihovně DLL. Knihovna DLL pro zařízení firmy ADVANTECH se jmenuje adsapi32.dll a část její *interface unit* driver.pas je dokumentována obrázkem 4.
Výhodou knihovny DLL je, že je dostatečně obecná a může být použita i pro komunikaci s různými zařízeními ADVANTECH. Tato univerzálnost však její používání komplikuje, neboť, jak je patrné i z obr. 4, vyžaduje dobrou znalost řídících datových struktur a funkcí knihovny. Studenti proto mají při návrhu vlastních programů možnost použít předpřipravené části programu, které programování měřicí karty mnohem zjednodušují. Je však nutné mít stále na zřeteli, že předpřipravené části nemohou plně pokrýt výhody knihovny DLL, a proto jejich použití nemusí být vždy vhodné. (Např. pro programování rychlých měření jsou v knihovně DLL připraveny speciální funkce a postupy.)

4. Programování karty v Borland Delphi pomocí objektu TCard.

Pro návrh svých vlastních programů je v praktiku připraveno vývojové prostředí Borland Delphi, které používá programovací jazyk *Object Pascal,* což je upravená varianta Pascalu pro návrh grafických programů pro operační systém MS Windows. Borland Delphi je založeno na modelu objektově orientovaného programování (OOP), při kterém uživatel používá (nebo i vytváří) tzv. objekty obsahující nejen datové položky (jako u strukturovaného typu záznam (record)), ale i procedury a funkce (tzv. metody), které odpovídají akcím, které tyto objekty mohou provádět. Speciální objekty typu TComponent (tzv. komponenty) mohou být zavedeny v prostředí Delphi do palety komponent, odkud mohou být snadno zkopírovány na formulář (okno vytvářeného programu). Výhodou komponent je, že k některým vlastnostem (datovým položkám) komponent na formuláři lze přistupovat už v době návrhu programu prostřednictvím Inspektoru objektů. Ostatní položky a metody jsou přístupné jen v době chodu programu.

Pro ovládání karty byla vytvořena třída (tj. typ objektu) TCard, který má zpřístupněné následující vlastnosti a metody:

`StartChannel: SmallInt;`

Index prvního vstupního kanálu, který má být měřen. K dispozici jsou indexy 0..15 při jednostranném (single-ended) zapojení.

`StopChannel: SmallInt;`

Index posledního vstupního kanálu, který má být měřen. Spolu s vlastností **`StartChannel`** udává interval měřených kanálů.

`Range[Index: Integer]: TRange read GetRange;`

Pole dvojic typu záznam, které obsahují minimum a maximum rozsahu měření napětí zvlášť pro každý vstupní kanál. Hodnoty jsou odvozeny od hodnot v poli

`Gain[Index: Integer]: SmallInt;`

Pole pro nastavení rozsahu měření napětí na jednotlivých kanálech. Do pole se zapisuje index rozsahu, který odpovídá příslušnému rozsahu podle

`Gains: TStrings read fGainStrings;`

Objekt obsahující textové popisky k jednotlivým rozsahům. Index řetězce (začínající od nuly) je shodný s indexem, který se zadává do pole Gain.

`InputChannel[Index: Integer]: Double;`

Pole obsahující naměřené hodnoty napětí (ve voltech). Hodnoty jsou již převedeny podle zvoleného rozsahu kanálu z celého čísla na reálnou hodnotu napětí. Čtení z kanálu funguje, je-li daný kanál aktivní. Před čtením další naměřené hodnoty je nutné volat funkci **`Acquire`**.

`OutputChannel[Index: Integer]: Single;`

Pole dvou reálných proměnných, do kterých zapisujeme hodnoty napětí, po nichž chceme, aby se objevily na výstupu. Zadávat můžeme hodnoty v intervalu 0… 10 V.

`procedure Prepare;`

Procedura konfiguruje nastavení měřicí karty. Tzn., že změny nastavení týkající se výběru aktivních kanálů a jejich rozsahů budou uplatněny teprve po zavolání procedury **`Prepare.`**

`function Acquire: Boolean;`

Funkce provádí vlastní měření s parametry nastavenými po zavolání procedury **`Prepare.`** Po vlastním měření je naměřenými hodnotami naplněno pole **`InputChannel.`**

Použití objektu třídy TCard si budeme demonstrovat na jednoduchých příkladech.

Příklad 1: Zobrazení seznamu rozsahů kanálů

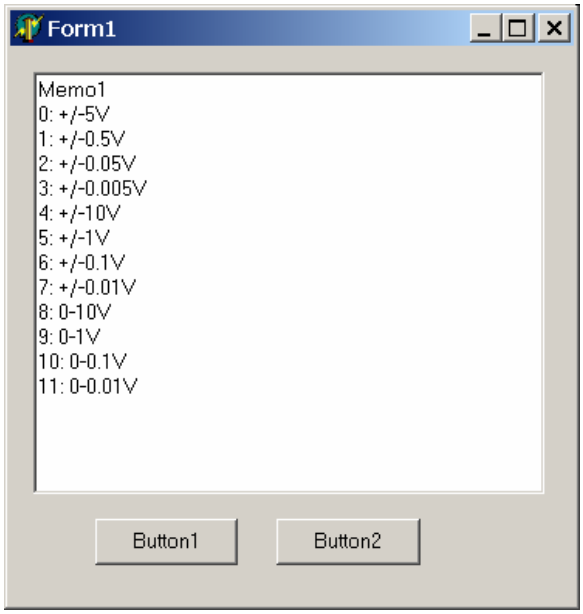
1. Otevřeme v Delphi nový projekt (*File → New application*).
2. Na formulář si dáme komponentu TCard, její jméno se automaticky nastaví na Card1. Od této chvíle můžeme ke kartě přistupovat pomocí objektu Card1.
3. Přidáme komponentu TMemo pro výpis textů (u nás to budou popisky rozsahů). Ikona komponenty TMemo je šestá ikona zleva v první záložce *Standard*.
   Jméno komponenty se automaticky nastaví na Memo1.
4. Na formulář přidáme tlačítko Button1 (sedmá ikona tamtéž). Dvojitým kliknutím na tlačítko se otevře okno a vytvoří se procedura, která se bude spouštět v době běhu programu při kliknutí na tlačítko. Mezi **`begin`** a **`end`** zapíšeme příkaz

   ```
   Memo1.Lines.AddStrings(Card1.Gains);
   ```

   který získaný seznam řetězců popisujících rozsahy vypíše do okna komponenty Memo1. Index rozsahů začíná nulo, tzn. První vypsaný rozsah má index `0`, poslední *celkový počet – 1* (`Memo1.Lines.Count-1`).
5. Program zkompilujeme a spustíme (klávesa F9). Při kliknutí na tlačítko se zobrazí seznam dostupných rozsahů.
6. Rozsahy můžeme i zobrazit např. následovně (reakce na stisknutí dalšího tlačítka `Button2`):

```
procedure TForm1.Button2Click(Sender: TObject);
var
  i: Integer;
begin
  for i:= 0 to Card1.Gains.Count-1 do
    Memo1.Lines.Add(IntToStr(i) + ': ' + Card1.Gains[i]);
end;
```

Tento postup způsobí vypsání rozsahů i s indexy rozsahů. Znalost indexu konkrétního rozsahu využijeme v příkladu 2. Výsledek tohoto postupu ukazuje Obrázek 5: Zobrazení dostupných rozsahů a jejich indexů.



Obrázek 5: Zobrazení dostupných rozsahů a jejich indexů.

Příklad 2: Změření hodnoty napětí na jednom kanálu

1. Zopakujeme kroky 1., 2. z příkladu 1.
2. Na formulář přidáme tlačítko (`Button1`) a panel pro vypsání měřené hodnoty (`Panel1`), který je reprezentován předposlední ikonou v záložce *Standard.*
3. Hodnotu budeme měřit při stisku tlačítka Button1. V obsluze kliknutí na tlačítko musíme nastavit shodný počáteční a koncový kanál (např. na nulu) a rozsah kanálu 0. Nastavení uplatníme příkazem `Prepare,` který je nutné zavolat i tehdy, měníme-li aktivní kanály v Inspektoru objektů. Měření se provede příkazem `Acquire`, který naplní nultý prvek pole `InputChannel`. Reálná hodnota v poli je převedena na řetězec funkcí `FloatToStr` a výsledný text je přiřazen popisku panelu:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  with Card1 do
  begin
    StartChannel:= 0; // aktivní kanál je
    StopChannel:= 0;  // pouze kanál č. 0
    Gain[0]:= 0;      // nastavení rozsahu kanálu 0
    Prepare;          // uplatni nastavení

    Acquire;          // změř napětí
    Panel1.Caption:= FloatToStr(InputChannel[0]) + ' V';
  end;
end;
```

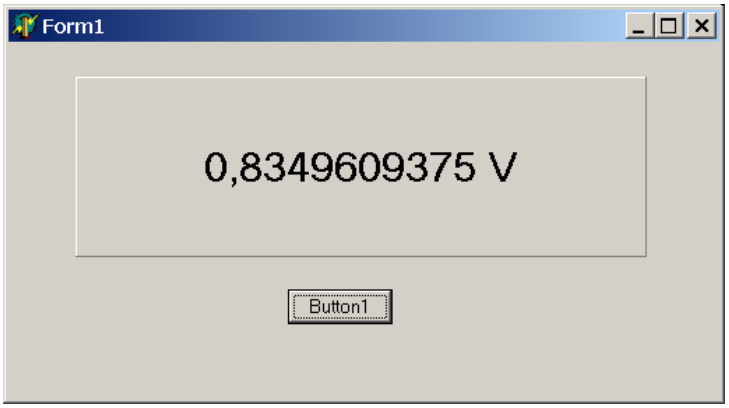
Aby nebylo při každém příkazu vztahujícím se ke kartě nutné zdlouhavě vypisovat "`Card1.`", je soubor příkazů obalen konstrukcí

`with` s čím **`do begin`** něco dělej **`end`**

Tlačítko samozřejmě můžeme tisknout opakovaně. Neměníme-li ale parametry měření, je výhodné nastavit je pouze jednou a dále opakovaně volat už jen

```
Acquire;
Panel1.Caption:= FloatToStr(InputChannel[0]) + ' V';
```

Výsledek je na Obrázek 6.



Obrázek 6: Změření a vypsání jedné hodnoty napětí.

Příklad 3: Opakované měření napětí na dvou kanálech

V tomto příkladu si ukážeme, jak měřit s určitým intervalem na dvou kanálech současně a výsledek vykreslovat do grafu.

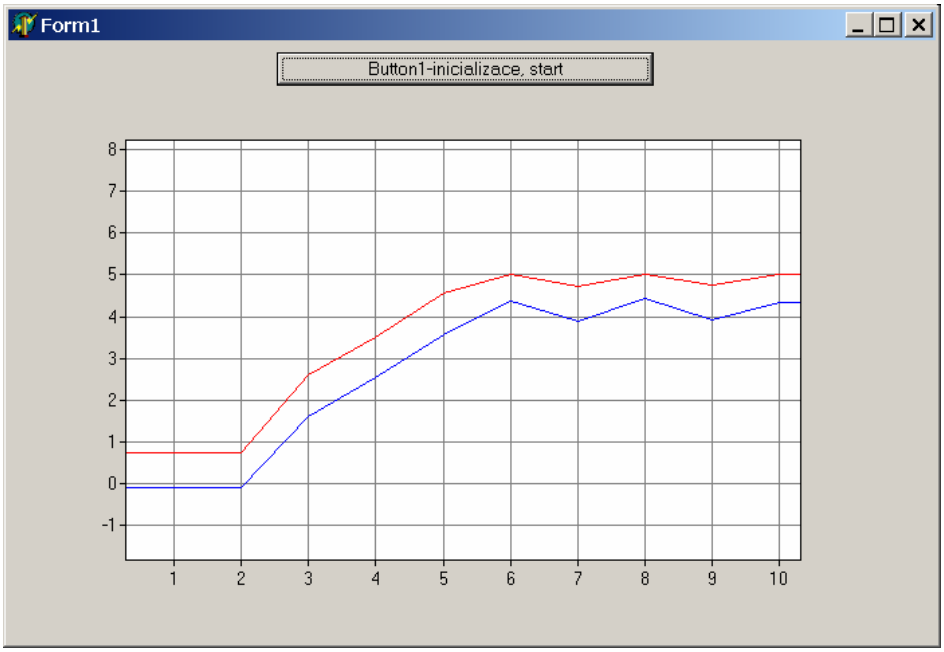
- Zdá se, že nejjednodušším způsobem, jak zajistit opakování měření, je použití příkazu `Acquire` v cyklu. Avšak užitím cyklu u složitějšího a časově náročnějšího měření se můžeme dočkat nepříjemného překvapení – program jakoby „tuhne", okno programu se nevykresluje, neboť v době běhu cyklu nejsou zpracovávány žádosti o jeho překreslení. Tento problém se profesionálně řeší použitím zvláštního procesu (tzv. vlákna), který má na starost jen měření, zatímco původní program naměřené hodnoty pouze vykresluje. Pro naše účely je ale naprosto dostačující komponenta `Timer` (první komponenta v záložce *System*), které se přiřadí procedura (opět dvojitým kliknutím nad komponentou na formuláři). Nastaví-li se u komponety `Timer` ještě vlastnost `Interval`, bude komponenta po uplynutí této doby v milisekundách opakovaně volat zadanou proceduru. Do procedury potom stačí zadat příkaz k měření a zobrazení dat. Je vhodné mít při spouštění programu `Timer` nejprve vypnut, což učiníme v Inspektoru objektů nastavením vlastnosti komponenty `Enabled` na `False`. Zapneme jej až po nastavení parametrů měření v obsluze inicializačního tlačítka:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Card1.StartChannel:= 0;
```

```
  Card1.StopChannel:= 1;
  Card1.Prepare;
  Timer1.Enabled:= True;
end;
```

- Pro zobrazení dat lze použít komponenty v záložce *Sgraph:*
  - `Tsp_XYPlot` je komponenta pro vykreslování grafu,
    - Nastavením příznaku BufferedDisplay na True lze odstranit případné blikání grafu. Graf je potom předem vykreslován na pomocnou bitmapu, která je poté už jen kopírována do okna programu.
  - `Tsp_XYLine` slouží pro uložení grafem zobrazovaných dat. Její důležité vlastnosti a metody:
    - `Plot` – graf, ve kterém se budou data vykreslovat
    - `LineAttr`, `PointAttr` obsahují parametry kreslené čáry a bodů
    - `AddXY(X,Y: Double)` – přidá další bod do seznamu, překreslí graf
    - `QuickAddXY(X,Y: Double)` – rychlé přidání bodu bez nového škálování os
    - `Clear` – vyčistí seznam bodů
    - `LockInvalidate: Boolean` – příznak, kterým lze v případě přidávání více bodů najednou dočasně zakázat neustále překreslování grafu po přidání každého bodu. Značně zrychlí proces přidávání bodů do grafu.

Hotový program může vypadat obdobně jako na Obrázek 7.



Obrázek 7: Vykreslování časového průběhu napětí na dvou kanálech.

- Pro uložení dat do textového souboru lze využít mírně modifikovaných funkcí standardního Pascalu

```
var
  Soubor: TextFile;
  X, Y: Double;
begin
  X:= 5.03; Y:= 1e-6;
  AssignFile(Soubor, 'jmeno_souboru.txt');
  ReWrite(Soubor);
  WriteLn(Soubor, X:10:2, Y:10:3);
  CloseFile(Soubor);
end;
```

nebo s výhodou vypisovat data do některé komponenty (např. TMemo) a poté využít její metodu pro uložení napsaného textu:

```
Memo1.Lines.Clear;
Memo1.Lines.Add(FloatToStr(X), ' ', FloatToStr(Y));
Memo1.Lines.SaveToFile('jmeno_souboru.txt');
```

Poznámka: Vytvořené programy jsou pod názvy `p1.dpr` až `p3.dpr` uloženy v adresáři C:\Vyuka\PCI1710HG\priklady. Před jejich zkoušením je nutné obsah adresáře překopírovat do studentského adresáře C:\Home\Student.