Parametrizace AtouchX

Abstrakt

Aplikační poznámka se zabývá ukázkou použití komunikačního ovladače AtouchX v prostředí Microsoft Visual C# Express a v prostředí Microsoft Excel.

Autor: Michal Kupčík, Zbyněk Říha
Dokument: ap0013_cz_04.pdf

Příloha

Obsah souboru: ap0013_cz_03.zip

ap0013_p01_cz_02.zip	Projekty pro řídicí systémy
ap0013_p02_cz_02.xls	Excel – čtení a zápis jednoduché proměnné
ap0013_p03_cz_02.xls	Excel – čtení a zápis matice
ap0013_p04_cz_02.xls	Excel – editace času v řídicím systému
ap0013_p05_cz_02.xls	Excel – čtení archívu
ap0013_p06_cz_02.xls	Excel – čtení provozního deníku
ap0013_p07_cz_02.xls	Excel – komunikace se dvěma identickými řídicími systémy v síti
ap0013_p08_cz_02.xls	Excel – aktivní komunikace řídicího systému s PC
ap0013_p09_cz_02.zip	C# – čtení a zápis jednoduché proměnné
ap0013_p10_cz_02.zip	C# – čtení a zápis matice
ap0013_p11_cz_02.zip	C# – editace času v řídicím systému
ap0013_p12_cz_02.zip	C# – čtení archívu
ap0013_p13_cz_02.zip	C# – čtení provozního deníku
ap0013_p14_cz_02.zip	C# – komunikace se dvěma identickými řídicími systémy v síti
ap0013_p15_cz_02.zip	C# – aktivní komunikace řídicího systému s PC
ap0013_p16_cz_01.zip	Delphi – ukázková aplikace, další potřebné informace

Obsah

Historie revizí

Verze	Datum	Autor změny	Změny
001	22. 06. 2009	Říha Zbyněk, Kupčík Michal	Nový dokument.
002	02. 01. 2013	Říha Zbyněk	Změna obrázků, sjednocení názvů textBoxu (ErrText) v kapitole 5.
003	11. 05. 2017	Kupčík Michal	Změna obrázků, doplnění kapitoly 5.1.
004	07. 02. 2018	Říha Zbyněk	V aplikacích pro Excel zrušena reference na objekt AtouchXArchive.

Související dokumentace

1. Nápověda k části PseDet vývojového prostředí DetStudio
   soubor: Psedet_cs.chm
2. Nápověda ke komunikačnímu ovladači AtouchX
   soubor: AtouchX.chm

1 Definice pojmů

DetStudio
Vývojové prostředí firmy AMiT, které slouží pro parametrizaci řídicích systémů. Toto prostředí je volně ke stažení na amitomation.cz.

Stanice
Řídicí systém nebo PC v síti DB-Net(/IP).

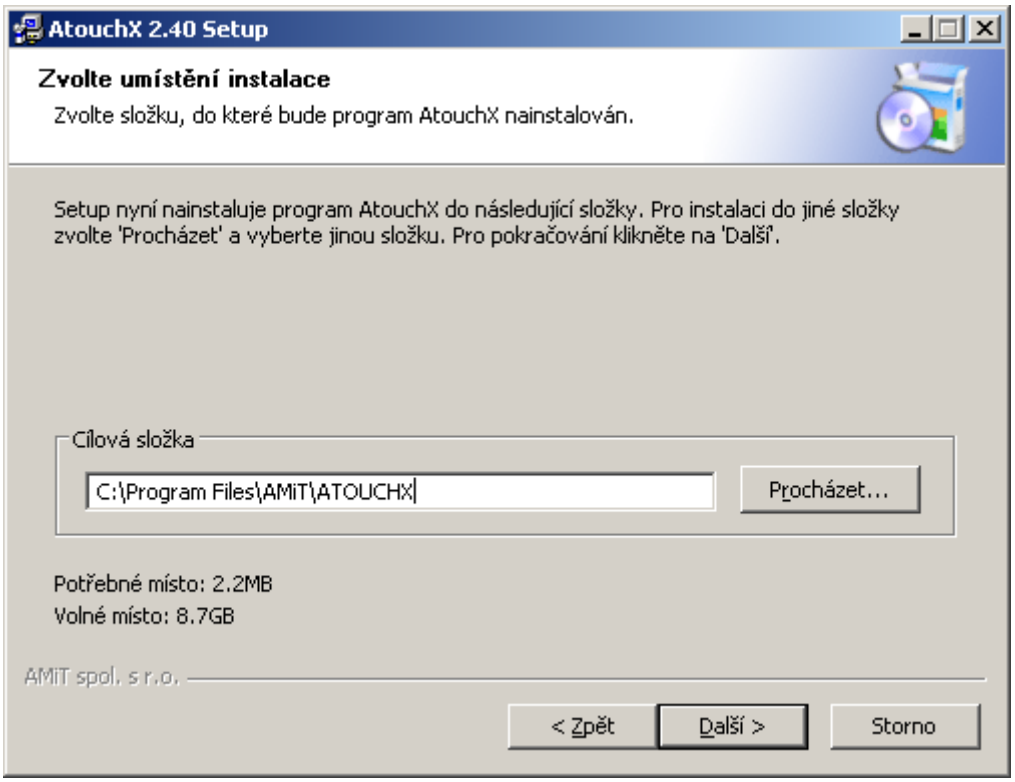
2 AtouchX

AtouchX je komunikační ovladač, který zajišťuje vazbu mezi řídicími systémy a aplikacemi na PC. AtouchX je ideální prostředek pro všechny programátory, kteří vyvíjejí vlastní aplikace na PC a potřebují zajistit přenos údajů z/do řídicích systémů firmy AMiT. Obsahuje několik ActiveX objektů, které jsou určeny pro plnohodnotný přístup k datům v řídicích systémech. Kromě základního přenosu dat mezi řídicími systémy a aplikací na PC podporuje také tzv. „zpětné archivy“, přenosy času a data, zajišťuje detekci stavu stanic a další funkce.

Všechny objekty knihovny AtouchX mají funkcionální charakter (rozhraní). V praxi to znamená, že objekty nemají žádné vlastnosti (nebo jen několik málo vlastností a to spíše provozního charakteru) a veškerá práce s nimi probíhá pomocí metod (funkcí).

2.1 Instalace AtouchX

Instalační soubor komunikačního ovladače AtouchX je volně ke stažení na amitomation.cz. Po spuštění instalace bude otevřen průvodce instalací, ve kterém zadáme cestu, kam mají být nainstalovány ukázkové aplikace a nápověda ke komunikačnímu ovladači. Knihovny, které AtouchX pro komunikaci využívá, jsou vždy instalovány do systémového adresáře Windows.



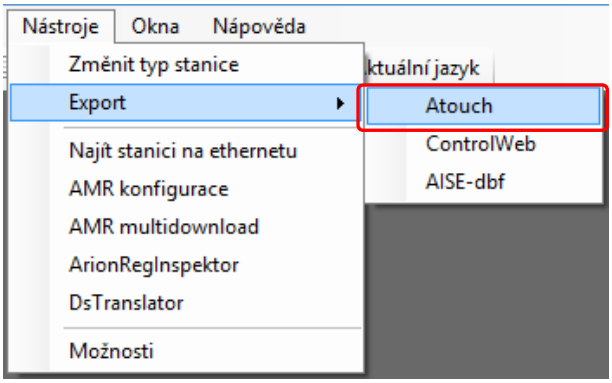
Obr. 1 – Instalace AtouchX

3 Parametrizace AtouchX

Parametrizaci komunikačního ovladače lze mimo jiné provést prostřednictvím tří *.ini souborů, pomocí kterých zadáme ovladači seznam proměnných řídicího systému, ze / do kterých chceme číst / zapisovat, rozhraní, prostřednictvím kterého chceme s řídicím systémem komunikovat (sériová linka, Ethernet, atd.) a popis archivů či provozního deníku. Struktura těchto souborů je blíže popsána v nápovědě ke komunikačnímu ovladači AtouchX.

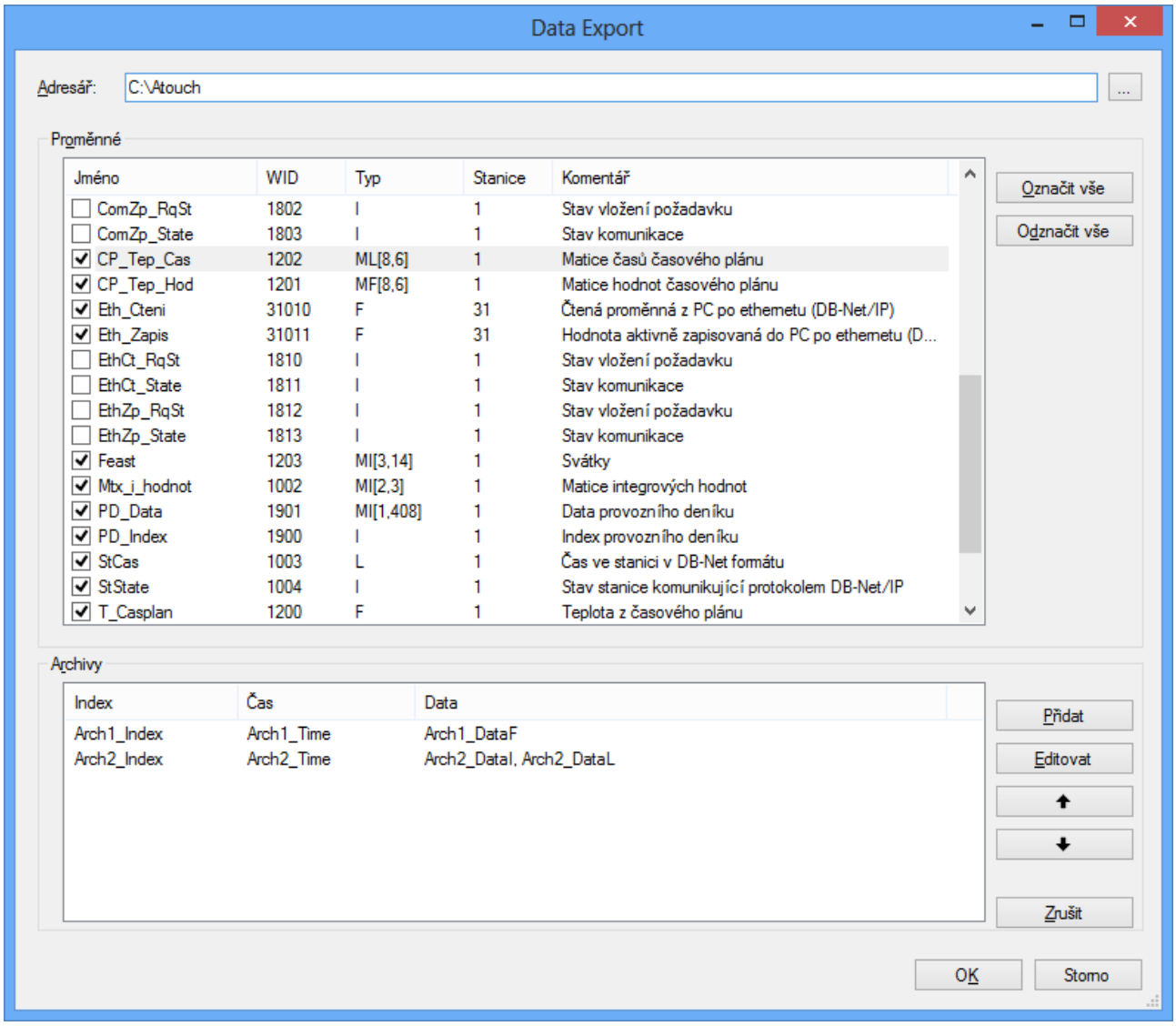
3.1 Export parametrizačních souborů z DetStudia

Parametrizační soubory vytvoříme přímo prostřednictvím návrhového prostředí DetStudio pomocí menu **Nástroje / Export / Atouch**.



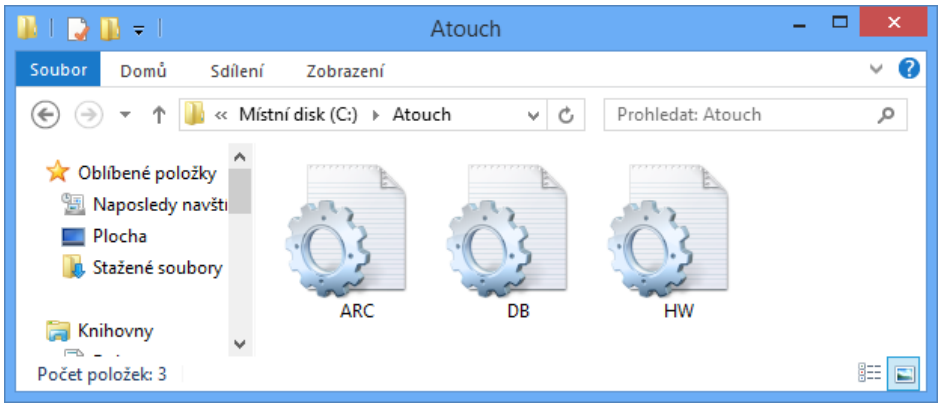
Obr. 2 – Export parametrizačních souborů

Výběrem položky **Atouch** dojde k otevření okna, ve kterém zadáme adresář, kam mají být parametrizační soubory exportovány. Vybereme proměnné, které mají být čteny / zapisovány a v případě požadavku na práci s archívem z řídicího systému nadefinujeme také archívy.



Obr. 3 – Výběr adresáře pro export parametrizačních souborů

Kliknutím na tlačítko „**OK**“ dojde k uzavření okna „Výběr seznamu proměnných ze seznamu databáze“ a k vytvoření dvou až tří parametrizačních souborů HW.ini, SW.ini a případně ARC.ini v námi zvoleném adresáři.



Obr. 4 – Vytvořené *.ini soubory

V souboru DB.ini je popis všech databázových proměnných, vybraných při exportu. V souboru HW.ini jsou zadány komunikační parametry pro spojení s řídicím systémem. V souboru ARC.ini je popis vybraných archivů a aplikačního provozního deníku. Komunikační parametry jsou generovány dle komunikace nastavené v návrhovém prostředí DetStudio.

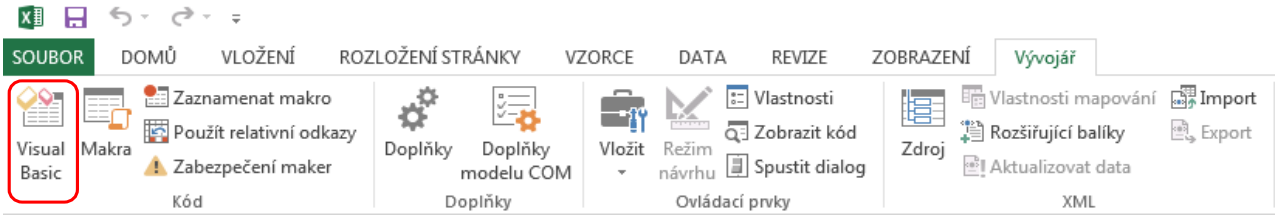
4 Ukázkové aplikace v Microsoft Excel

V příloze této aplikační poznámky lze nalézt ukázkové projekty vytvořené v tabulkovém editoru Microsoft Excel 2013.

4.1 Nastavení Microsoft Excel 2013

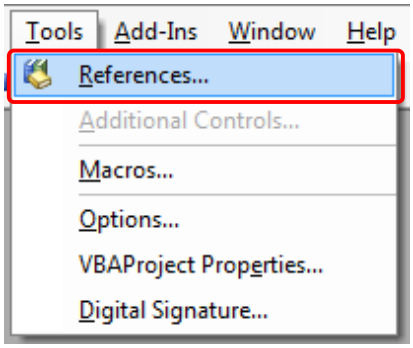
Pro práci s ovladačem AtouchX je nutné mít v prostředí Microsoft Excel aktivovánu práci s makry a musí být umožněno spouštění maker.

Knihovnu AtouchX lze v Excelu využívat pouze ve spojení s programovacím jazykem Visual Basic. Pomocí tlačítka **Visual Basic** v nabídce **Vývojář** tedy přejdeme do editoru jazyka Visual Basic.



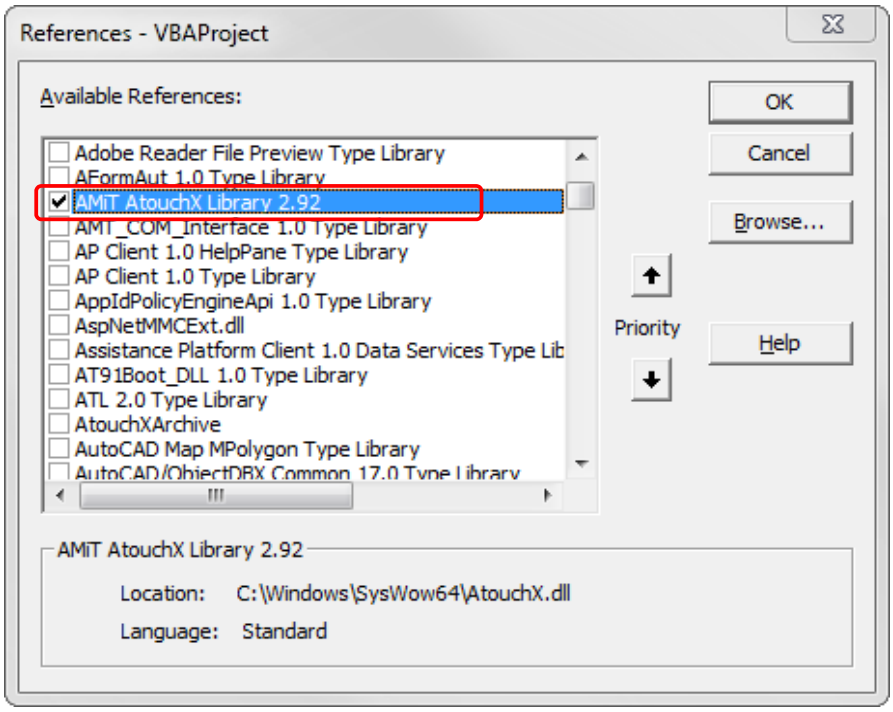
Obr. 5 – Přechod do editoru jazyka Visual Basic

V editoru jazyka Visual Basic musíme Excelu oznámit, že budeme pracovat s knihovnou AtouchX, což učiníme pomocí menu **Tools / References**.



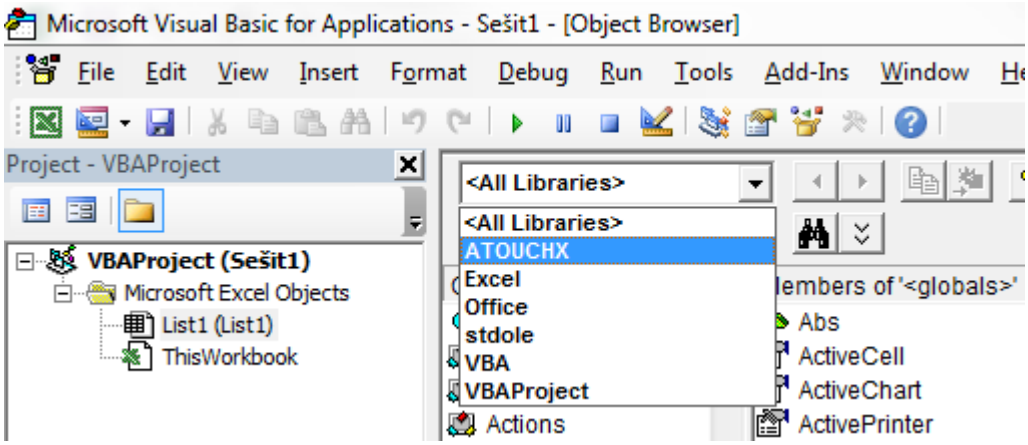
Obr. 6 – Otevření okna s referencemi

Dojde k otevření okna „References – VBA project“, ve kterém vyhledáme a zvolíme příslušnou knihovnu.



Obr. 7 – Volba potřebných knihoven

Možnosti knihovny AtouchX můžeme nyní zjistit pomocí okna „Object Browser“, které lze otevřít z menu **View / Object Browser**.



Obr. 8 – Zobrazení seznamu objektů knihovny AtouchX

Po výběru knihovny ATOUCHX se v okně „Object Browser“ zobrazí seznam objektů knihovny. Popis těchto objektů lze nalézt v nápovědě, která se instaluje společně s komunikačním ovladačem AtouchX.

4.2 Tvorba aplikace

Ve stromu projektu dvakrát klikneme levým tlačítkem myši např. na položku „List1 (List1)“. Otevře se zatím prázdná plocha formuláře, do které můžeme psát kód programu.

4.2.1 Definice AtouchApp

V ukázkových aplikacích budeme využívat pro komunikaci objekt „AtouchApp“. Ze všeho nejdříve musíme nadefinovat objekt typu „AtouchApp“. Toto učiníme např. následujícím kódem

```
Public WithEvents ATC As AtouchApp
```

4.2.2 Inicializace připojení k síti DB-Net (DB-Net/IP)

Pro inicializaci vytvoříme makro, které nazveme „AtouchInicializace“.

```
Public Sub AtouchInicializace()
EndSub
```

V tomto makru nejdříve vytvoříme instanci komunikačního ovladače pomocí příkazu „New“ (vytvoříme objekt s názvem „ATC“).

```
Set ATC = New AtouchApp
```

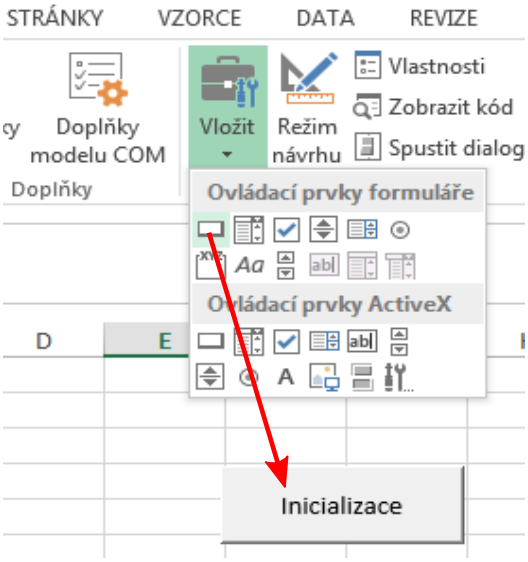
Pro vlastní inicializaci připojení k síti DB-Net (DB-Net/IP) použijeme např. metodu „InitFromFile“. Tato metoda provede inicializaci připojení k síti pomocí dvou externích souborů. V našem případě nesou název hw.ini a db.ini. Metoda také vrací kód chyby. Tento kód budeme ukládat do globální proměnné „Retrn“ typu Integer.

```
Retrn = ATC.InitFromFile(ActiveWorkbook.Path & "\hw.ini", ActiveWorkbook.Path &
"\db.ini")
```

Dále zjistíme, zda inicializace proběhla úspěšně či neúspěšně. V případě, že došlo k chybě inicializace, otevřeme okno s chybovým hlášením a zrušíme objekt „ATC“. Pokud inicializace proběhla v pořádku, zapíšeme do buňky B4 text „Inicializace OK.“

```
If Retrn <> arrOK Then
     MsgBox "Nepovedlo se napojení na DBNet. " & vbCrLf & "Číslo chyby " &
Hex$(Retrn), vbOKOnly
     ATC.Done
     Exit Sub
Else
     Me.Cells(4, 2) = "Inicializace OK"
End If
```

Do tabulky MS Excel vložíme tlačítko, změníme jeho text na „Inicializace“ a přiřadíme mu námi vytvořené makro.



Obr. 9 – Vložení tlačítka

Po vypnutí režimu návrhu dojde stiskem tlačítka k inicializaci připojení k síti DB-Net (DB-Net/IP).

4.2.3 Ukázka metody, která nevyvolává událost

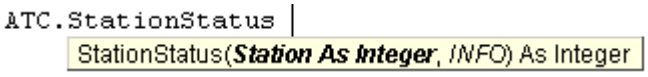
Jako ukázkovou metodu, která nevyvolává událost, zvolíme metodu „StationStatus“ (zjištění stavu připojení stanice). Opět vytvoříme makro, které nazveme „StationStatus“.

```
Public Sub StationStatus()
End Sub
```

V manuálu se dočteme, že syntaxe pro metodu „StationStatus“ je následující:

```
objekt.StationStatus (ByVal Station As Integer, ByRef INFO As Variant) As Integer
```

V našem případě používáme objekt „ATC“. Nejjednodušší zjištění, jak má vypadat správný zápis, je použití plovoucí nápovědy, která se automaticky objevuje při psaní příkazů.



Obr. 10 – Plovoucí nápověda

Nadefinujeme si proměnnou „Station“ typu Integer, ve které bude číslo stanice, jejíž stav chceme zjistit a proměnnou „Info“ typu Variant.

Do námi vytvořeného makra pak vložíme následující kód:

```
Station = INT(1)                          'uložení hodnoty 1 do proměnné Station
Retrn = ATC.StationStatus(Station, Info)  'volání události
```

Dále zjistíme, zda byla metoda zpracována úspěšně či neúspěšně. V případě, že byla zpracována neúspěšně, zobrazíme okno s chybovým hlášením. V případě, že byla zpracována úspěšně, uložíme stav a typ HW připojení do buněk C4 a C5.

```
If Retrn <> arrOK Then
     MsgBox "Chyba zjištění stavu. " & vbCrLf & "Číslo chyby " & Hex$(Retrn), vbOKOnly
     Exit Sub
Else
     Me.Cells(4, 3) = Info(0)              'uložení typu HW připojení do buňky C4
     Me.Cells(5, 3) = Info(1)              'uložení stavu připojení do buňky C5
EndIf
```

Do tabulky MS Excel opět vložíme tlačítko. Jeho text změníme na „Status“ a přiřadíme mu námi vytvořené makro. Po úspěšné inicializaci připojení k síti DB-Net bude po jeho stisku zpracován kód, který makro obsahuje.

4.2.4 Ukázka metody, která vyvolává událost

Jako ukázku metody, která vyvolává událost, zvolíme metodu „NetGetData“, která vyvolává událost „EndNetGetData“.

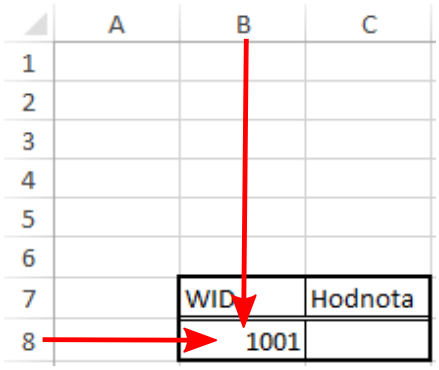
Vytvoříme makro, pomocí kterého provedeme příkaz pro načtení hodnoty proměnné. Makro nazveme „GetData“.

```
Public Sub GetData()
End Sub
```

Pro vyvolání požadavku na čtení dat využijeme metodu „NetGetData“ jejíž syntaxe je následující:

```
objekt.NetGetData (ByVal WID As Long, ByVal Param As Long) As Integer
```

WID proměnné, jejíž hodnotu chceme číst z řídicího systému, budeme zadávat v tabulce MS Excel z buňky B8.



Obr. 11 – Buňka pro zadání hodnoty WIDu

Nadefinujeme si proměnnou „WID“ typu Integer a proměnnou „Param“ typu Long. Do námi vytvořeného makra pak vložíme následující kód:

```
WID = Me.Cells(8, 2)                  'načtení hodnoty WIDu z buňky B8 do proměnné WID
Retrn = ATC.NetGetData(WID, Param)    'vyvolání události
Me.Cells(2, 3) = Retrn                'uložení výsledku volání události do buňky C2
```

Metoda přečte obsah proměnné, jejíž WID jsme zadali do buňky B8 (čtení probíhá asynchronně) a po ukončení čtení je vyvolána událost „EndNetGetData“ jejíž syntaxe je následující.

```
Private Sub objekt_EndNetGetData ([index As Integer], ByVal WID As Long, ByVal Result
As Long, ByVal Param As Long, ByVal DATA As Variant)
```

Metoda „EndNetGetData“ oznámí výsledek komunikace a hodnotu proměnné s požadovaným WIDem.

```
Private Sub ATC_EndNetGetData(ByVal WID As Long, ByVal Result As Long, ByVal Param As
Long, ByVal DATA As Variant)
```

Nejprve zjistíme výsledek komunikace. Pokud komunikace skončila chybou, zobrazíme okno s příslušným chybovým hlášením. Pokud komunikace skončila úspěšně, uložíme hodnotu čtené proměnné do buňky C8.

```
If ((Result And atfMaskstate) <> atfOk) Then
     MsgBox "Komunikace se nezdařila. " & vbCrLf & "Číslo chyby " & Hex$(Result And
atfMaskstate), vbOKOnly
  Else
     Me.Cells(8, 3) = DATA
  End If
End Sub
```

Do tabulky MS Excel opět vložíme tlačítko. Jeho text změníme na „Čti Data“ a přiřadíme mu makro „GetData“. Po úspěšné inicializaci připojení k síti DB-Net bude po jeho stisku do buňky C8 načtena hodnota skalární proměnné s WIDem 1001.

4.2.5 Ukončení komunikace

Stejně jako jsme inicializovali připojení k síti DB-Net (DB-Net/IP), musíme toto připojení také správně ukončit. Pro ukončení připojení k síti DB-Net (DB-Net/IP) slouží metoda „Done“, jejíž syntaxe je následující:

```
objekt.Done () As Integer
```

Vytvoříme makro, které nazveme „Konec“.

```
Public Sub Konec()
End Sub
```

V makru pak provedeme kontrolu, zda objekt „ATC“ existuje a v případě, že ano, tak jej zrušíme a uvolníme paměť. Výsledný kód, který do makra vložíme, tedy bude vypadat následovně:

```
If Not (ATC Is Nothing) Then ATC.Done     'jestli objekt existoval, tak je zrušíme
Set ATC = Nothing                         'uvolníme paměť
```

Do tabulky MS Excel vložíme tlačítko. Jeho text změníme na „Konec“ a přiřadíme mu makro „Konec“.

4.2.6 Práce s archívy

Pro práci s archívy je nutné vytvořit objekt typu „AtouchArch“. Tento vytvoříme stejně jako objekt „AtouchApp“ v kapitole 4.2.1 „Definice AtouchApp“.

```
Public WithEvents ATCA As AtouchArch
```

Pozor!
Objekt pro svou činnost potřebuje, aby současně s ním existoval a pracoval některý z objektů poskytujících připojení k síti DB-Net (DB-Net/IP).

Inicializace archívu

Pro inicializaci vytvoříme makro, které nazveme „ArchivInicializace“.

```
Public Sub ArchivInicializace()
End Sub
```

V tomto makře nejdříve vytvoříme instanci archívu pomocí příkazu „New“.

```
Set ATCA = New AtouchArch
```

Pro vlastní inicializaci archívu použijeme např. metodu „InitFromFile“. Tato metoda provede inicializaci archívu pomocí externího souboru. V našem případě jej nazveme pd_arch.ini (struktura je popsána v nápovědě ke komunikačnímu ovladači AtouchX). Metoda také vrací kód chyby. Tento kód budeme ukládat do globální proměnné „Retrn“ typu Integer.

```
Retrn = ATCA.InitFromFile(ActiveWorkbook.Path & "\pd_arch.ini")
```

Hodnotu proměnné „Retrn“ zobrazíme v buňce tabulky MS Excel.

```
Me.Cells(2, 3) = Retrn
```

Do tabulky MS Excel vložíme tlačítko. Jeho text změníme na „Arc Init“ a přiřadíme mu námi vytvořené makro. Po úspěšné inicializaci připojení k síti DB-Net bude po jeho stisku inicializován archiv a výsledek inicializace bude vložen do buňky C2.

Povolení chodu archívu

Pro povolení chodu archívu vytvoříme makro, které nazveme „ArcChod“.

```
Public Sub ArcChod()
End Sub
```

V naší aplikaci budeme využívat automatický archív (bližší informace viz nápověda ke komunikačnímu ovladači AtouchX). Pro povolení chodu archívu využijeme metodu „Control“ jejíž syntaxe je následující:

```
objekt.Control (ByVal AID As Integer, ByVal Run As Boolean) As Integer
```

V makru vytvoříme proměnnou „AID“ typu Integer a „Run“ typu Boolean. Kód, který do makra vložíme, pak bude vypadat následovně:

```
AID = 0                                 'číslo archívu (viz nápověda k AtouchX)
Run = True                              'Povolení chodu archívu
RetrnArc = ATCA.Control(AID, Run)       'Metoda pro povolení chodu archívu
```

Do tabulky MS Excel vložíme tlačítko. Jeho text změníme na „Arc Start“ a přiřadíme mu námi vytvořené makro. Po úspěšné inicializaci archívu se stiskem tohoto tlačítka povolí jeho chod.

Uložení vzorku archívu

Pro uložení vzorku archívu slouží událost „Sample“. Tato je vyvolána, je-li k dispozici jeden vzorek automatického archivu. Její syntaxe je následující:

```
Private Sub objekt_Sample ([index As Integer], ByVal AID As Integer, ByVal DATA As
Variant)
```

Nadefinujeme tedy událost „Sample“ včetně obsluhy přijetí vzorku. Kód pak bude vypadat následovně:

```
Private Sub ATCA_Sample(ByVal AID As Long, ByVal DATA As Variant)
Dim SampleOK As Boolean                 'Definice proměnné typu Boolean
Dim AcceptOK As Integer                 'Definice proměnné typu Integer

Me.Cells(Radek, 1) = DATA(0)            'Uložení času vzorku do buňky
Me.Cells(Radek, 2) = DATA(1)            'Uložení archivní hodnoty do buňky
Radek = Radek + 1                       'Přesun na další řádek
SampleOK = True                         'Vzorek vždy přijmeme
AcceptOK = ATCA.Accept(AID, SampleOK)   'Potvrzení příjmu vzorku
End Sub
```

V události „Sample“ je nutné využít také metodu „Accept“ pro přijetí nebo odmítnutí vzorku. Popis této metody lze nalézt v nápovědě ke komunikačnímu ovladači AtouchX.

Ukončení činnosti objektu AtouchArch

Stejně jako jsme archív inicializovali, musíme jej také správně ukončit. Pro ukončení slouží metoda „Done“, jejíž syntaxe je následující:

```
objekt.Done () As Integer
```

Ukončení objektu AtouchArch vložíme např. do stejného makra jako ukončení připojení k síti DB-Net (DB-Net/IP). V makru pak provedeme kontrolu, zda objekt „ATCA“ existuje a v případě, že ano, tak jej zrušíme a uvolníme paměť. Výsledný kód, který do makra vložíme, tedy bude vypadat následovně:

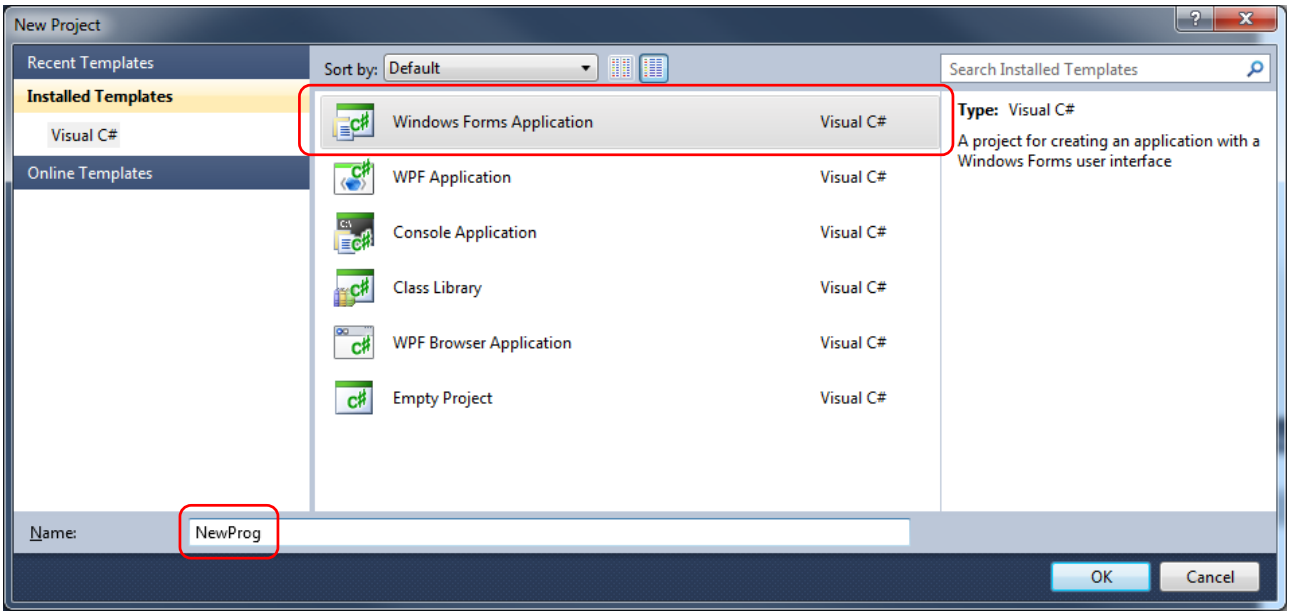
```
If Not (ATCA Is Nothing) Then ATCA.Done   'jestli objekt existoval, tak je zrušíme
Set ATCA = Nothing                        'uvolníme paměť
```

5 Ukázkové aplikace v Microsoft Visual C# Express

V příloze této aplikační poznámky lze nalézt ukázkové projekty vytvořené v prostředí Microsoft Visual C# Express.

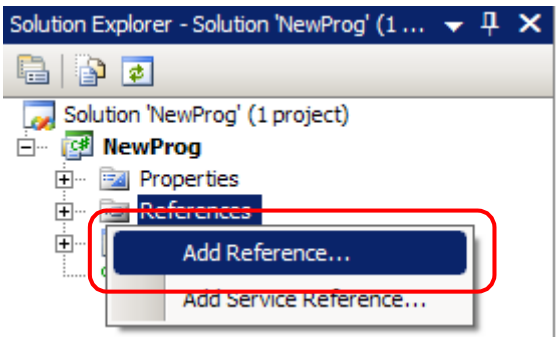
5.1 Nastavení Visual C# Express

Po spuštění programu vybereme z hlavního menu položku **File / New Project**. Otevře se okno „NewProject“, ve kterém vybereme položku **Windows Form Application** a změníme jméno vytvářeného projektu.



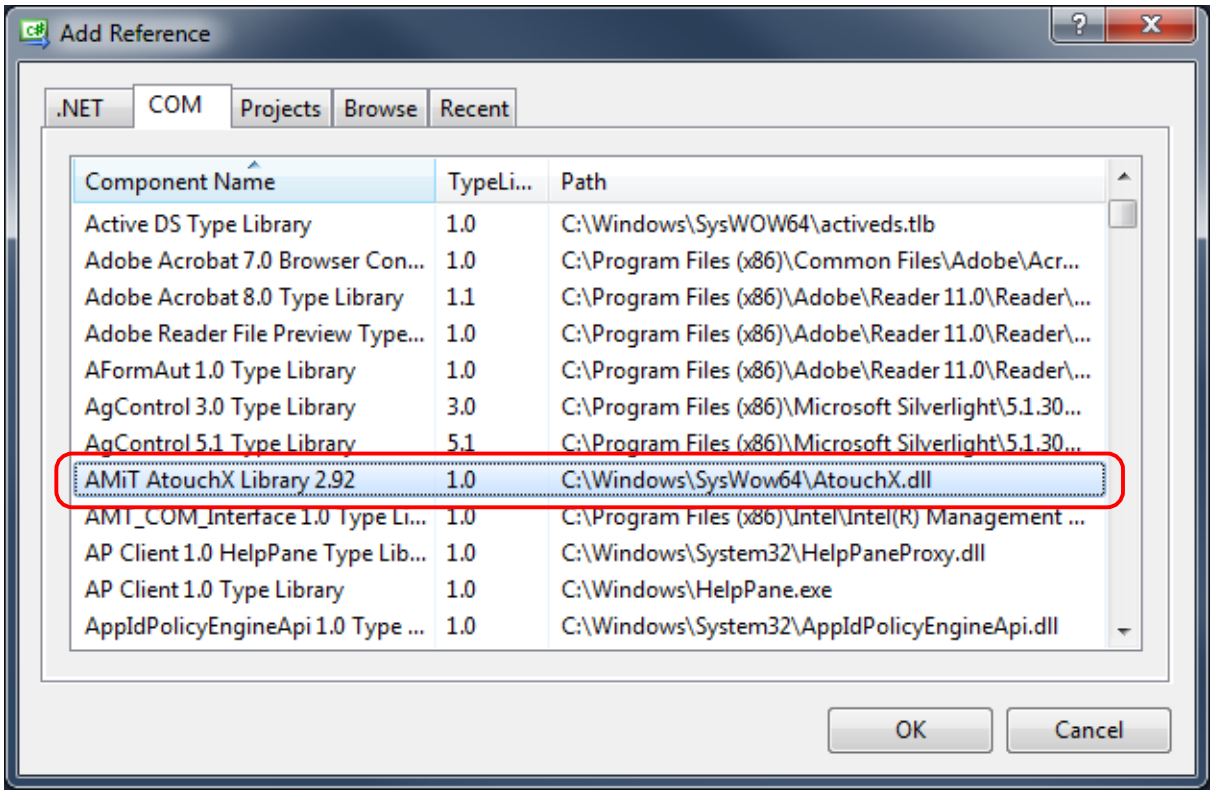
Obr. 12 – Založení projektu ve Visual C# Express

Abychom mohli v prostředí Visual C# Express využívat knihovnu AtouchX, musíme nejdříve v okně „Solution Explorer“ přidat referenci na příslušné ActiveX prvky. Toto lze učinit kliknutím pravého tlačítka myši na položku **Reference** a následně výběrem položky **Add reference**.



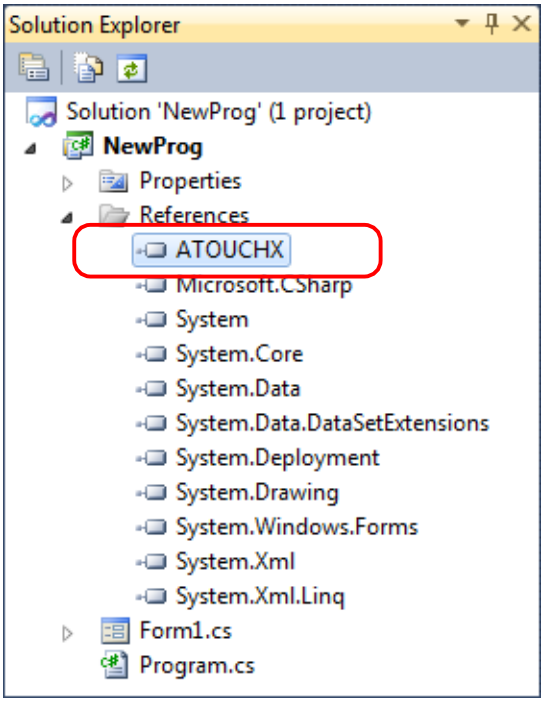
Obr. 13 – Otevření okna s referencemi

Dojde k otevření okna „Add Reference“, ve kterém vyhledáme a zvolíme příslušnou knihovnu v záložce „COM“ dle následujícího obrázku.



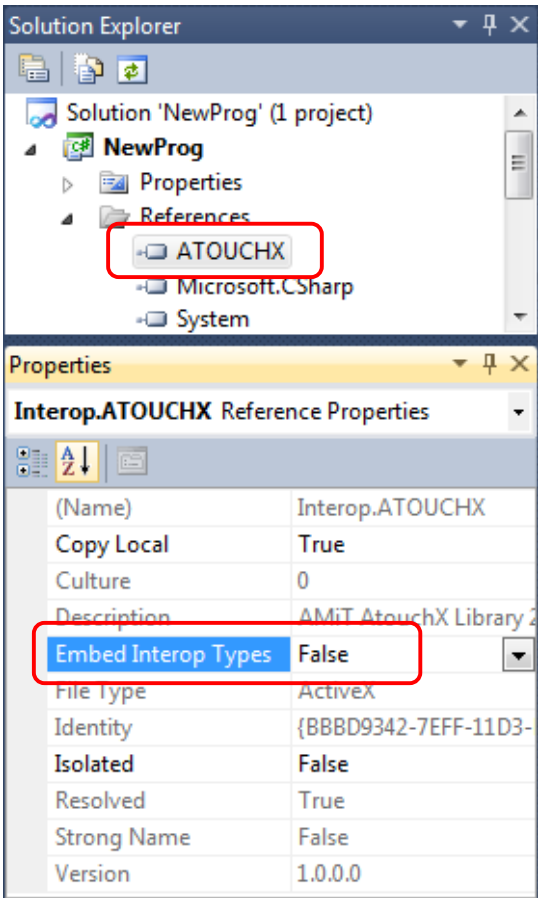
Obr. 14 – Volba potřebných knihoven

Korektní přidání reference ověříme rozvinutím položky **References** v okně „Solution Explorer“. Zde nalezneme položku AtouchX.



Obr. 15 – Reference na AtouchX byla korektně vložena

Aby byly všechny funkce knihovny AtouchX v uživatelské aplikaci zcela funkční, je potřeba nastavit této knihovně vlastnost **Embed Interop Types** na hodnotu **False**.



Obr. 16 – Změna hodnoty vlastnosti Embed Interop Types po označení knihovny AtouchX

5.2 Tvorba aplikace

Dvakrát klikneme na zatím ještě prázdnou plochu formuláře. Tím se dostaneme do okna, ve kterém lze psát kód programu.

Ze všeho nejdříve umístíme kurzor myši na začátek kódu pod použití slova „using“. Zde zapíšeme:

```
using ATOUCHX;
```

Výsledný kód tedy bude vypadat dle následujícího obrázku.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ATOUCHX;
```

Obr. 17 – Použití using

5.2.1 Definice AtouchApp

Pro práci s objektem „AtouchApp“ je potřeba vytvořit jeho instanci pomocí příkazu „new“ a to již v části kódu „public partial class Form1 : Form“. Do této části, která vznikne po dvojkliku na vytvořený formulář, se rovněž zapisují deklarace dalších globálních proměnných a objektů.

Kód pro vytvoření instance objektu „AtouchApp“ tedy bude vypadat následovně:

```
AtouchApp App = new AtouchApp();
```

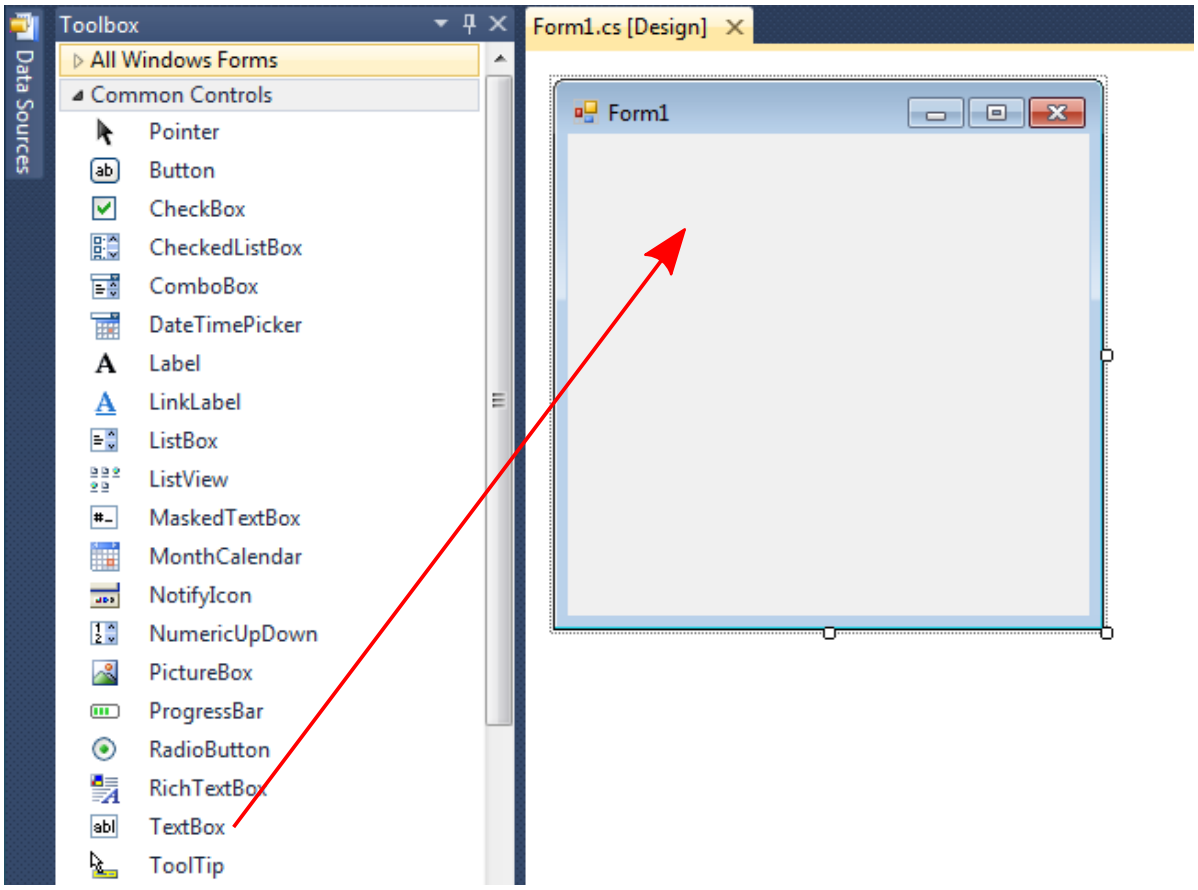
```
namespace NewProg
{
    public partial class Form1 : Form
    {
        AtouchApp App = new AtouchApp();
        public Form1()
        {
            InitializeComponent();
        }
```

Obr. 18 – Vytvoření instance objektu AtouchApp

5.2.2 Inicializace připojení k síti DB-Net (DB-Net/IP)

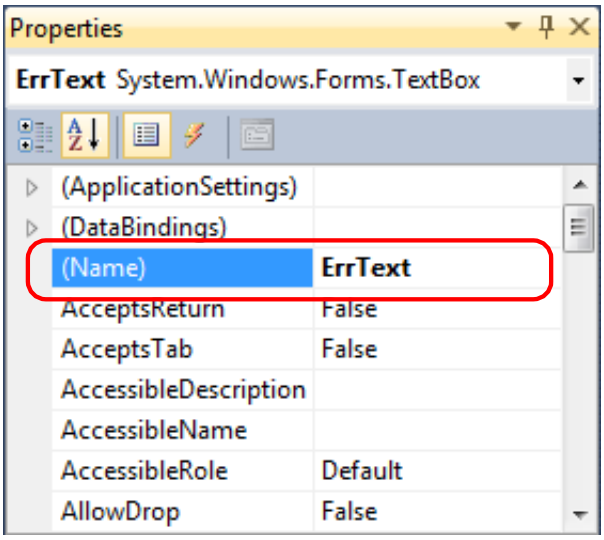
Inicializaci provedeme v části kódu „Form1_Load“. Pro vlastní inicializaci připojení k síti DB-Net (DB-Net/IP) použijeme např. metodu „InitFromFile“. Tato metoda provede inicializaci připojení k síti pomocí dvou externích souborů. V našem případě nesou název hw.ini a db.ini. Metoda také vrací kód chyby. Tento kód budeme ukládat do globální proměnné „Err“ typu Integer a její hodnotu vypíšeme na formulář.

Nejdříve se přepneme zpět do návrhu designu formuláře a v okně „Toolbox“ vybereme prvek „TextBox“.



Obr. 19 – Umístění prvku TextBox na formulář

Umístíme jej na formulář a v okně „Properties“ jej v položce „Name“ pojmenujeme např. „ErrText“.



Obr. 20 – Přejmenování prvku TextBox

Kód chyby, který vrací metoda „InitFromFile“ je ve tvaru čísla. Prvek „TextBox“ však zobrazuje na formuláři text (typ String). Proto proměnnou „Err“ nejprve přetypujeme pomocí metody „ToString()“. Výsledný kód pak bude vypadat následovně:

```
private void Form1_Load(object sender, EventArgs e)
{
    //Inicializační metoda pro komunikaci přes síť DB-Net.
    App.VariantOnly = false;
    Err = App.InitFromFile("hw.ini", "db.ini");
    ErrText.Text = Err.ToString();
}
```

Obr. 21 – Inicializace AtouchApp a výpis výsledku inicializace na formulář

5.2.3 Ukázka metody, která nevyvolává událost

Jako ukázkovou metodu, která nevyvolává událost, zvolíme metodu „StationStatus“ (zjištění stavu připojení stanice).

V manuálu se dočteme, že syntaxe pro metodu „StationStatus“ je následující:

```
objekt.StationStatus (ByVal Station As Integer, ByRef INFO As Variant) As Integer
```

V našem případě používáme objekt „App“. Nejjednodušší zjištění, jak má vypadat správný zápis, je použití plovoucí nápovědy, která se automaticky objevuje při psaní příkazů.

```
App.StationStatus(
    short IAtouchApp.StationStatus(short Station, ref object INFO)
```

Obr. 22 – Plovoucí nápověda

Z plovoucí nápovědy je zřejmé, že z VBA typů Integer a Variant se stal v C# typ Short a Object. Nadefinujeme si proto dvě proměnné (ať už globální nebo lokální) typu Short a Object, které pojmenujeme např. „ShoStation“ a „ObjInfo“.

```
short ShoStation = 1;      //zjišťujeme stav řídicího systému s číslem 1
object ObjInfo = null;
```

Z okna „Toolbox“ umístíme na formulář tlačítko. Pojmenujeme jej např. „ShowStat“. Metodu pro zjištění stavu stanice budeme spouštět po kliknutí na toto tlačítko.

Metoda „StationStatus“ vrací taktéž kód chyby. Tento budeme vypisovat do již vytvořeného prvku „TextBox“, který jsme pojmenovali „ErrText“.

Kód vložený do události stisku tlačítka pak bude vypadat následovně:

```
Err = App.StationStatus(ShoStation,ref ObjInfo);
```

Po vykonání tohoto příkazu se změní obsah a typ proměnné „ObjInfo“. V nápovědě ke komunikačnímu ovladači AtouchX se dočteme, že se změní v jednorozměrnou matici s velikostí dvě buňky.

V následujícím kroku tedy nejdříve otestujeme, zdali se tak skutečně stalo. To provedeme podmínkou testující typ proměnné:

```
if (ObjInfo is ushort[])
```

Testujeme na typ ushort[], neboť víme, že hodnoty které můžeme obdržet, jsou v rozsahu 0 až 65535. Do těla této podmínky pak napíšeme zbývající část kódu.

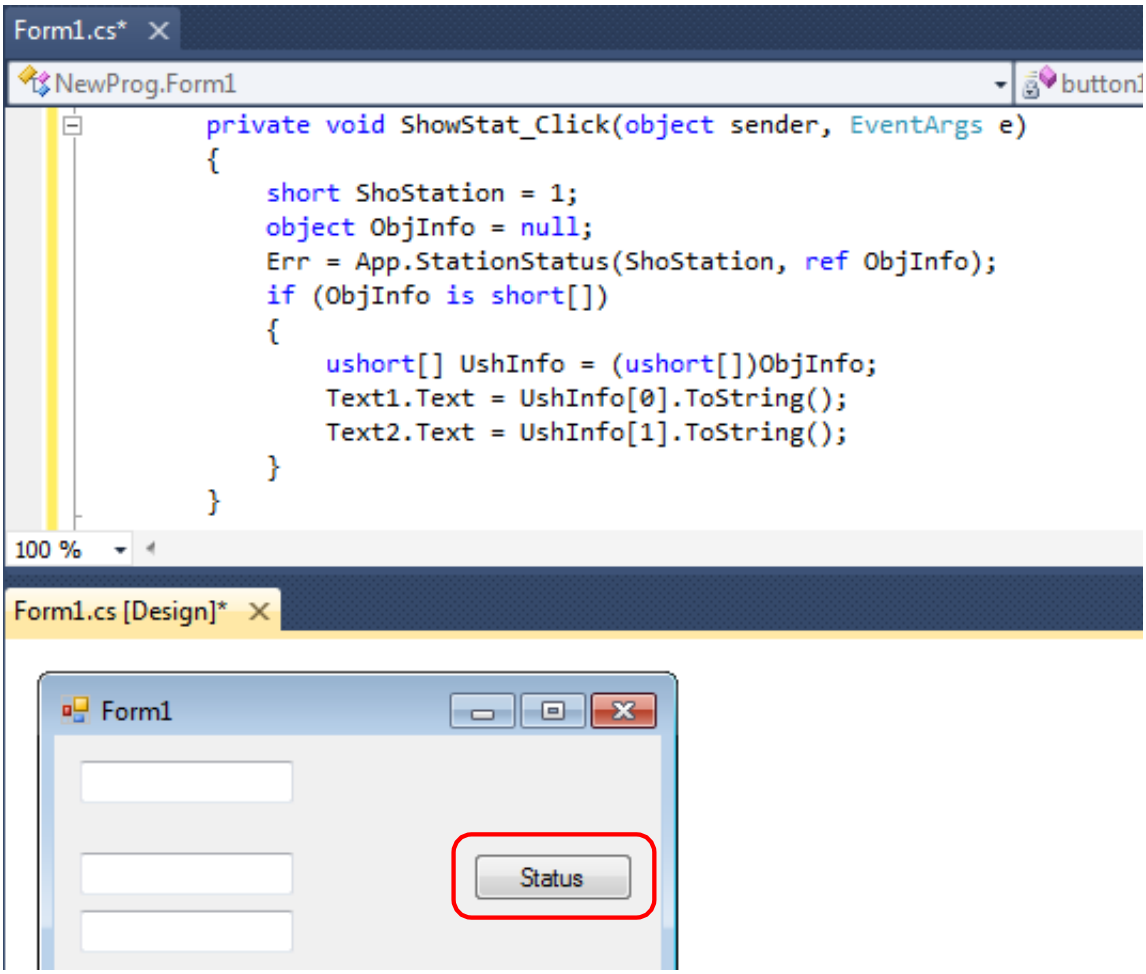
Jako první musíme vytvořit novou proměnnou typu ushort[] např. „UshInfo“, do které uložíme proměnnou „ObjInfo“, kterou musíme přetypovat pomocí výrazu (ushort[]).

```
ushort[] UshInfo = (ushort[])ObjInfo;
```

Pro zobrazení obsahu obou prvků matice na formulář umístíme další dva prvky typu „TextBox“. Pojmenujeme je „Text1“ a „Text2“. Pomocí jejich vlastnosti „Text“ do těchto dvou prvků budeme zapisovat hodnoty matice, převedené na typ String metodou „.ToString“.

```
Text1.Text = UshInfo[0].ToString();
Text2.Text = UshInfo[1].ToString();
```

Výsledný kód pak bude vypadat dle následujícího obrázku.



Obr. 23 – Kód volaný v události stisku tlačítka

5.2.4 Ukázka metody, která vyvolává událost

Jako ukázku metody, která vyvolává událost, zvolíme metodu pro čtení proměnné „NetGetData“, která vyvolává událost „EndNetGetData“.

Syntaxe metody „NetGetData“ je následující:

```
objekt.NetGetData (ByVal WID As Long, ByVal Param As Long) As Integer
```

Již známým postupem vytvoříme na formuláři tlačítko a do události stisku tohoto tlačítka vložíme následující kód:

```
Err = App.NetGetData(1001, 1);
ErrText.Text = Err.ToString();
```

Tím jsme naprogramovali načtení hodnoty proměnné s WIDem 1001. Výsledek volání této metody je uložen do proměnné „Err“, kterou opět zobrazujeme v prvku „TextBox“ s názvem „ErrText“.

Událost „EndNetGetData“, kterou metoda „NetGetData“ vyvolává je potřeba „zapnout“. Toto učiníme tak, že do části programu s hlavičkou „public Form1()“ napíšeme název události, znaménka „+=“ a poté dvakrát stiskneme klávesu TAB. Tím se automaticky doplní text pro inicializaci události a vytvoří se tělo funkce, která se spustí při vyvolání události.

```
public Form1()
{
    InitializeComponent();
    App.EndNetGetData += new _IAtouchAppEvents_EndNetGetDataEventHandler(App_EndNetGetData);
}

void App_EndNetGetData(int WID, int Result, int Param, object DATA)
{
    throw new NotImplementedException();
}
```

Obr. 24 – Vytvoření události

Řádek začínající „throw“ vymažeme a v budoucnu místo něj zapíšeme kód, který se má vykonat, pokud se událost vyvolá. Jelikož pracujeme s událostí, která vrací hodnotu proměnné, předpokládá se, že tuto hodnotou chceme nadále určitým způsobem zpracovat. Pokud by se jednalo o zpracování numerické nebo jakékoliv jiné, při kterém se nevyžaduje, aby se načtená hodnota proměnné vypsala na formulář, píšeme toto zpracování přímo do připraveného obslužného těla události. Pokud však bude potřeba načtenou hodnotu přímo zobrazit na formuláři, pouhé napsání

```
TextBoxX.Text = DATA.ToString();
ReadData.Text = DATA.ToString();
```

k výsledku nevede. Po spuštění této části kódu by se vždy objevila chyba, která je způsobena rozdělením aplikace na více vláken. Jedno zajišťuje práci s formulářem a jiné obsluhuje události. Výše popsaným kódem tak dojde ke kolizi vláken.
Abychom zjištěnou hodnotu vypsali na formulář, musíme postupovat následovně:

1. Vytvoříme definici delegáta v místě globálních proměnných. V jeho argumentech jsou zapsány všechny proměnné, se kterými budeme chtít nadále pracovat. V našem případě to bude WID čtené proměnné, výsledek komunikace, jeden doplňující parametr a načtená data.
   Syntaxe pro zápis je následovná:

   ```
   private delegate void (návratová hodnota, v našem případě se nic nevrací)
   DelEndGetData (námi zvolený název) (int WID, int Result, int Param, object DATA)
   (předávané proměnné)
   ```

   Definice delegáta v našem případě tedy bude vypadat následovně:

   ```
   private delegate void DelEndGetData(int WID, int Result, int Param, object DATA);
   ```

2. Použijeme metodu „this.Invoke“, která má jako argumenty delegáta z předchozího řádku a následně seznam předávaných proměnných.

   ```
   this.Invoke(new DelEndGetData(App_EndNetGetData), new Object[] { WID, Result,
   Param, DATA });
   ```

   Kód události běží v jiném vlákně, než které zabezpečuje vykreslovací okno, proto je potřeba pomocí příkazu „invoke“ spustit kód pro práci s oknem do vykreslovacího vlákna.

3. Na základě výsledku komunikace vypíšeme na formulář načtenou hodnotu nebo chybový kód. V případě, že komunikace dopadla dobře, zjistíme (např. pomocí parametru „Param“) jakou proměnnou jsme komunikovali a v závislosti na hodnotě „Param“ zapíšeme načtenou hodnotu do příslušného textového pole.

Výsledný kód pak bude dle následujícího obrázku.

```
public partial class Form1 : Form
{
    //Vytvoření delegáta pro komunikaci mezi vlákny
1.  private delegate void DelEndGetData(int WID, int Result, int Param, object DATA);

    //Nutná definice objektů a proměnných.
    AtouchApp App = new AtouchApp();
    int Err;
    Object Data = new Object();

    public Form1()
    {
        InitializeComponent();
        //Inicializace návratových událostí komunikace. Po napsání App.xxx += a dvojím
        //stisku klávesy Tab se automaticky vytvoří obslužná těla událostí.
        App.EndNetGetData += new _IAtouchAppEvents_EndNetGetDataEventHandler(App_EndNetGetData);
    }

    void App_EndNetGetData(int WID, int Result, int Param, object DATA)
    {
2.      if (this.InvokeRequired)
        {
            //Kód je spuštěn v jiném vlákně, než které zabezpečuje vykreslování okna, proto
            //je potřeba pomocí příkazu invoke spustit kód pro práci s oknem do vykreslovacího vlákna.
            this.Invoke(new DelEndGetData(App_EndNetGetData), new Object[] { WID, Result, Param, DATA });
            return;
        }
        //Zde již kód běží ve vykreslovacím vlákně a je možné ovlivňovat zobrazovací prvky
        //Zápis na formulář po zpracování požadavku na čtení proměnné.
3.      if ((Result & (Int16)AtouchX_Communication_State.atfOk) != 0)
        {   //Komunikace skončila OK.
            ComText.Text = "OK";
            switch (Param)
            {
                case 1:
                    ReadInt.Text = DATA.ToString();
                    break;
                case 2:
                    ReadLon.Text = DATA.ToString();
                    break;
                case 3:
                    ReadFlo.Text = DATA.ToString();
                    break;
            }
        }
        else
        {   //Komunikace skončila chybou.
            ComText.Text = Result.ToString();
        }
    }
```

Obr. 25 – Tři kroky pro výpis hodnoty na formulář

5.2.5 Ukončení komunikace

Stejně jako jsme inicializovali připojení k síti DB-Net (DB-Net/IP), musíme toto připojení také správně ukončit. Pro ukončení připojení k síti DB-Net (DB-Net/IP) slouží metoda „Done“, jejíž syntaxe je následující:

```
objekt.Done () As Integer
```

Vytvoříme tlačítko, které nazveme „Konec“. Do události stisku tohoto tlačítka vložíme následující kód:

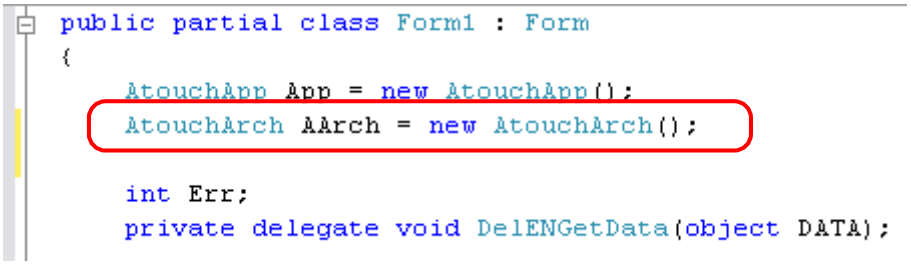
```
App.Done();
this.Close();
```

Tím je komunikace ukončena.

5.2.6 Práce s archívy

Pro práci s archívy je nutné vytvořit instanci objektu typu „AtouchArch“. Tuto vytvoříme stejně jako instanci objektu „AtouchApp“ v kapitole 5.2.1 „Definice AtouchApp“.

```
AtouchArch AArch = new AtouchArch();
```



Obr. 26 – Vytvoření instance objektu AtouchArch

Pozor!
Objekt pro svou činnost potřebuje, aby současně s ním existoval a pracoval některý z objektů poskytujících připojení k síti DB-Net (DB-Net/IP).

Inicializace archívu

Pro vlastní inicializaci archívu použijeme např. metodu „InitFromFile“. Tato metoda provede inicializaci archívu pomocí externího souboru. V našem případě jej nazveme PD_Arch.ini (struktura je popsána v nápovědě ke komunikačnímu ovladači AtouchX). Metoda také vrací kód chyby. Tento kód budeme ukládat do globální proměnné „Err“ typu Integer.

```
Err = AArch.InitFromFile("pd_arch.ini");
```

Hodnotu proměnné „Err“ zobrazíme v prvku „TextBox“ s názvem „ArcErr“, který jsme umístili na formulář.

```
ArcErr.Text = Err.ToString();
```

Výsledný kód inicializace archívu pak bude vypadat následovně.

```
private void Form1_Load(object sender, EventArgs e)
{
    //Inicializační metoda pro komunikaci přes síť DB-Net.
    Err = App.InitFromFile("hw.ini", "db.ini");
    if (Err == (Int16)AtouchX_Error_Code.arrOK)
    {
        //Inicializační metoda pro zavedení popisů archívu
        Err = AArch.InitFromFile("PD_Arch.ini");
    }
    ErrText.Text = Err.ToString();
}
```

Obr. 27 – Inicializace archívu

Tento objekt potřebuje ke svému plnohodnotnému použití nadefinování událostí „Completed“ a „Sample“. Proto vepíšeme do části programu „public Form1()“ kód „NázevObjektu.NázevUdálosti +=“ a dvakrát stiskneme klávesu TAB.

```
public Form1()
{
    InitializeComponent();
    AArch.Sample += new _IAtouchArchEvents_SampleEventHandler(AArch_Sample);
    AArch.Completed += new _IAtouchArchEvents_CompletedEventHandler(AArch_Completed);
}
```

Obr. 28 – Definice událostí pro AtouchArch

Do kódu události „Sample“ později vložíme metodu „Accept“, aby se po příjmu hodnoty vzorek potvrdil / zamítnul a mohl se přijmout vzorek další. Do kódu události „Completed“ později vložíme metodu „Control“ pro ukončení chodu archívu.

Povolení chodu archívu

V naší aplikaci budeme využívat automatický archív (bližší informace viz nápověda ke komunikačnímu ovladači AtouchX).

Pro povolení chodu archívu využijeme metodu „Control“ jejíž syntaxe je následující:

```
objekt.Control (ByVal AID As Integer, ByVal Run As Boolean) As Integer
```

Na formulář umístíme tlačítko a do události stisku tohoto tlačítka vložíme následující kód:

```
Err = AArch.Control(0, true); //Povolení chodu archívu
ArcErr.Text = Err.ToString(); //Zobrazení kódu, který metoda vrací
```

Stiskem tlačítka tak povolíme chod archívu s AID 0 (viz nápověda ke komunikačnímu ovladači AtouchX).

Uložení vzorku archívu

Pro uložení vzorku archívu slouží událost „Sample“. Tato je vyvolána, je-li k dispozici jeden vzorek automatického archivu. Tuto událost jsme nadefinovali již v podkapitole „Inicializace archívu“ a je nutné v ní využít také metodu „Accept“ pro přijetí nebo odmítnutí vzorku. Popis této metody lze nalézt v nápovědě ke komunikačnímu ovladači AtouchX.

```
AArch.Accept(0, true); //Přijetí vzorku z archivu AID = 0 (viz nápověda AtouchX)
```

Výsledný kód celé události tedy bude vypadat dle následujícího obrázku.

```
void AArch_Sample(int AID, object DATA)
{
    if (DATA is object[])
    {
        //Zpracování přijatých dat
    }
    AArch.Accept(0, true);
}
```

Obr. 29 – Potvrzení přijetí nového vzorku

Ukončení chodu archívu

Ukončení chodu archívu lze provést v události „Completed“, která je vyvolána v případě, kdy již archív neobsahuje žádné nové vzorky, které ještě nebyly načteny. Tuto událost jsme nadefinovali již v podkapitole „Inicializace archívu“. Pro ukončení chodu archívu použijeme již zmíněnou metodu „Control“. Výsledný kód celé události pak bude vypadat dle následujícího obrázku.

```
void AArch_Completed(int AID, int Samples)
{
    AArch.Control(0, false);
}
```

Obr. 30 – Ukončení chodu archívu

Ukončení činnosti objektu AtouchArch

Stejně jako jsme archív inicializovali, musíme jej také správně ukončit. Pro ukončení slouží metoda „Done“, jejíž syntaxe je následující:

```
objekt.Done () As Integer
```

Ukončení objektu AtouchArch vložíme např. do události stisku stejného tlačítka jako ukončení připojení k síti DB-Net (DB-Net/IP). Výsledný kód, který události vložíme, tedy bude vypadat následovně:

```
AArch.Done();    // ukončení činnosti AtouchArch
App.Done();      // ukončení činnosti AtouchApp
this.Close();    // zavření formuláře
```

6 DODATEK A

6.1 Konverze typů proměnných v C#

Jedná se o problematiku, která může nastat např. při použití metod „AtouchApp“, které zapisují hodnoty (proměnných, času) směrem z PC do řídicích systémů. V manuálu se dočteme, že některé metody (např. „NetPutData“ a „NetPutTime“) posílají hodnoty dle VBA v proměnných typu variant. V C# je ekvivalentní typ Object.

Aby řídicí systém dokázal zpracovat vstupní data a přenos komunikace neskončil chybou, musí být hodnota v této objektové proměnné vhodně konvertována. K tomuto nejlépe poslouží metoda „Convert“.

Typ Integer v řídicím systému má velikost 16 bitů, proto použijeme konverzi ve tvaru:

```
Convert.ToInt16(zdrojová_proměnná).
```

Podobně typ Long zkonvertujeme pomocí:

```
Convert.ToInt32(zdrojová_proměnná).
```

Typ Float v řídicím systému má v C# obdobu typ Single:

```
Convert.ToSingle(zdrojová_proměnná).
```

Příklad

Na formuláři máme prvek „TextBox1“ a v programu proměnnou „Data“ typu Object. Naším cílem je přečíst číslo z prvku „TextBox1“ a tuto hodnotu zapsat v řídicím systému do proměnné typu Integer s WIDem 1001. Použijeme tedy následující zápis:

```
Data = Convert.ToInt16(TextBox1.Text);
Err = App.NetPutData(1001, 1, Data);
```

Další typ, jenž vyžaduje konverzi je formát času. Ve VBA i v C# je to typ DateTime. Chceme-li přečíst čas z prvku na formuláři, kde je zobrazen v typu String, opět použijeme metodu „Convert“, nyní s funkcí „ToDateTime(zdrojová_proměnná)“.

Příklad

Chceme synchronizovat čas v řídicím systému s časem na PC. Synchronizace času řídicího systému s PC proběhne tehdy, pokud do řídicího systému zapíšeme čas „1.1.1980 0:00:00“.

Řešení:

```
DateTime DTCas = Convert.ToDateTime(TextBox1.Text);
Err = App.NetPutTime(1, 1, DTCas);
```

Kde v prvku „TextBox1“ bude uložen řetězec 1.1.1980 0:00:00.

7 DODATEK B

7.1 AtouchX v Delphi

Postup zprovoznění AtouchX v Delphi Standard verze 5.00 (Build 5.62).

Nejprve je třeba „naučit“ Delphi znát korektně objekt „AtouchApp“. Toho dosáhneme příkazem **Projekt / ImportTypeLibrary**. Objeví se nám okno, v jehož horní polovině je seznam všech registrovaných ActiveX knihoven. Neregistrované knihovny se dají přidávat pomocí tlačítka **Add**. Pokud jsme knihovnu AtouchX správně nainstalovali, objeví se v tomto seznamu "AMiT AtouchX Library 1.0", který vybereme. Ve spodní části okna ponecháme vybranou volbu "Generate Component Wrapper" a tlačítkem **CreateUnit** okno uzavřeme. Tím Delphi v podadresáři „Imports“ vytvoří soubor ATOUCHX_TLB.PAS, který "zapouzdří" všechny ActiveX objekty z knihovny do objektů Delphi.

Při generaci však vznikne malý problém. Objekty knihovny AtouchX používají parametry se jmény „Type“, „Set“ a „Result“. „Type“ a „Set“ jsou klíčová slova Delphi a při konverzi je Delphi doplní zepředu podtržítkem. Ovšem „Result“ není klíčové slovo, takže ho Delphi ponechá jak je. Jméno „Result“ se však používá pro označení návratové hodnoty funkcí a tak dochází ke konfliktu jmen. Problém je třeba odstranit ručně. Upravený soubor ATOUCHX_TLB.PAS je součástí souboru ap0013_p16_cz_xx.zip. Jeho nakopírováním do podadresáře „Imports“ by Delphi mělo začít znát objekty z knihovny AtouchX. Pozor, jde o generovaný soubor, který se při opakovaném importování či obnovení znovu přegeneruje, čímž se ručně provedené změny („Result“ => „_Result“) ztratí.

Po importu máme k dispozici typ objektu „TAtouchApp“. Pokud si tedy vytvoříme jeho instanci, můžeme volat jeho služby, ale ještě nejsme schopni zachytávat jeho události, přestože objekt má vše připraveno pro jejich použití. Objekt totiž obsahuje privátní proměnné se jmény typu FOnXXX (například „FOnEndReqIdentify“), které mají obsahovat ukazatele na funkce vyvolávané při příchodu událostí. Podle definice jsou proměnné typu "funkce objektu" (viz například deklaraci typu „TAtouchEndReqIdentify“), takže pro použití je třeba provést následující:

Vytvoříme objekt potomka „TAtouchApp“. V tomto objektu nadefinujeme funkci s prototypem shodným s událostní funkcí a při konstrukci objektu zajistíme přiřazení funkce do proměnné FOnXXX . Pokud do dané funkce vepíšeme svůj kód, vyvolá se při příchodu události.

Pro snadnější pochopení je součástí souboru ap0013_p16_cz_xx.zip také zdrojový kód, na kterém je propojení vyzkoušeno (viz soubor UNIT1.PAS).

Definujeme objekt „TMyAtouchAp“ jako potomka „TAtouchApp“. Předefinujeme konstruktor „Create“ a definujeme novou funkci „ENI“ (jako „EndNetIdentify“). Konstruktor „Create“ volá konstruktor předka a pak jen přiřadí "událostní funkci" do "událostní proměnné". Funkce „ENI“ pak zpracovává příchozí událost – kromě krátkého pípnutí zkontroluje, zda událost oznamuje úspěšné čtení identifikace a pokud ano, zobrazí přečtená data. Při vytváření hlavního formuláře pak vytvoříme objekt „ATC“ typu „TMyAtouchApp“ a inicializujeme jej (použijeme úplné cesty, které musíme přepsat dle své lokace) pomocí testovacího NULL připojení. Při rušení formuláře pak rušíme objekt „ATC“.

Ukázková aplikace pro Delphi, včetně dalších potřebných informací je v souboru ap0013_p16_cz_xx.zip, který je součástí přílohy této aplikační poznámky.

8 Technická podpora

Veškeré informace ohledně použití komunikačního ovladače AtouchX, Vám poskytne oddělení technické podpory firmy AMiT. Technickou podporu můžete kontaktovat nejlépe prostřednictvím emailu na adrese **support@amit.cz**.

9 Upozornění

AMiT spol. s r. o. poskytuje informace v tomto dokumentu, tak jak jsou, nepřejímá žádné záruky, pokud se týče obsahu tohoto dokumentu a vyhrazuje si právo měnit obsah dokumentu bez závazku tyto změny oznámit jakékoli osobě či organizaci.

Tento dokument může být kopírován a rozšiřován za následujících podmínek:

1. Celý text musí být kopírován bez úprav a se zahrnutím všech stránek.
2. Všechny kopie musí obsahovat označení autorského práva společnosti AMiT, spol. s r. o. a veškerá další upozornění v dokumentu uvedená.
3. Tento dokument nesmí být distribuován za účelem dosažení zisku.

   V publikaci použité názvy produktů, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.