NGS Framework

www.promax.it

User Guide

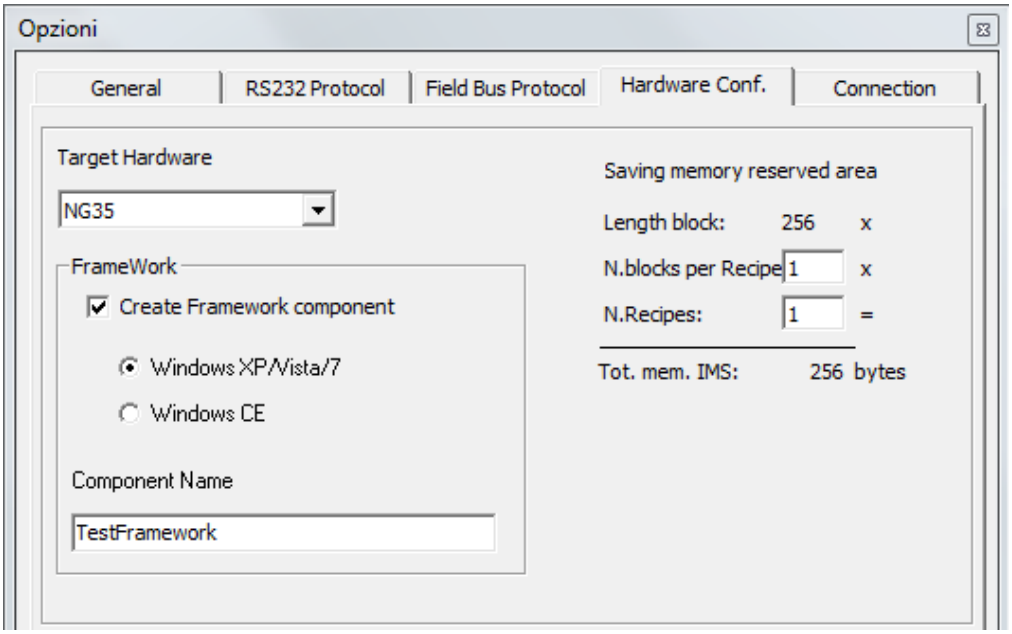
Rev. 2.00.0

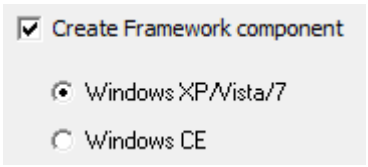
1. GENERAL DESCRIPTION

NGS Framework is a component created by VTB that is used on systems development. NET and compatible. The component provides a set of properties and methods that allow a simple and intuitive interface for hardware platforms NG35, NGM13 and compatible.
Basically using the development environment Microsoft Visual Studio 2005 or higher, you can create Windows XP or Windows CE that integrate the resources of a system Promax NGS.
The result is to get the flexibility of the products. NET combined with the power of the systems NGS.

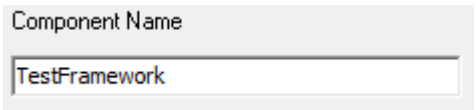
2. HOW CREATE A FRAMEWORK COMPONENT

To create a component for use in environments Visual Studio, you must configure VTB to enable the creation of componentframework.
This is done from the Menu Options





Check this option
and select the PC S.O.
Windows Xp,7,8
or Windws CE



Insert a Component (DLL) Name

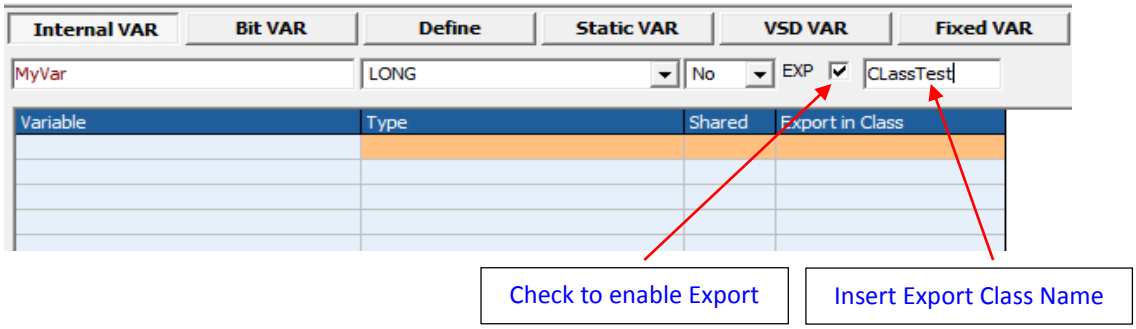
Then you need to decide on the variables and functions to be exported in the component framework.
There are the following restrictions:

WARNING

The variables must be global.
You can not export variables Bit but their only source variable.
You can only export internal variables static and fixed
They can only be exported functions of the MAIN page (global)

3. Variable Export

In the statement of VTB, EXP, enable the check box and enter the name of the class in which you want to export the variable. It is not obligatory to enter the name of the class, but this simplifies the grouping of variables organizing them by type, field of use, etc.. Only the structures, are exported to a class that takes the name of the variable declared in VTB.



4. Function Export

To export, you must declare the function in the following mode:

function TestFramework(Par1 as long) as long $_EXPORT_$ ClassFTest
.....
Endfunction

$_EXPORT_$	Keyword that identifies the Export function in the framework
ClassFTest	Class export function (field not required, if omitted, the function is exported to the class Generic)

WARNING

You should always give a different name for the classes of variable export and that of functions. As in example above, the variable has been exported to the class ClassTest and function in a different class ClassFTest.
You can export more variables or functions in the same class

5. System Variables Export

All the System Variables are automatically exported in the ***SystemVar*** Class

6. System Functions Export

All the System Functions are automatically exported in the following Class:

BoardNg*.Class*

Where *Class*:

CanOpen	CanOpen Functions
Hardware	Hardware resource functions
Interpola	Obsolete Class not use

WARNING

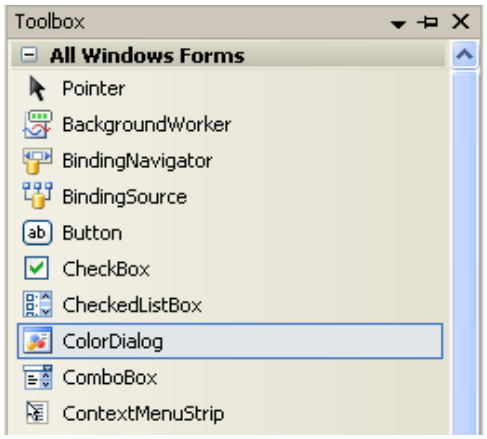
All system functions are contained in files that can be dynamically updated by REALUPDATE application.

7. Create a component

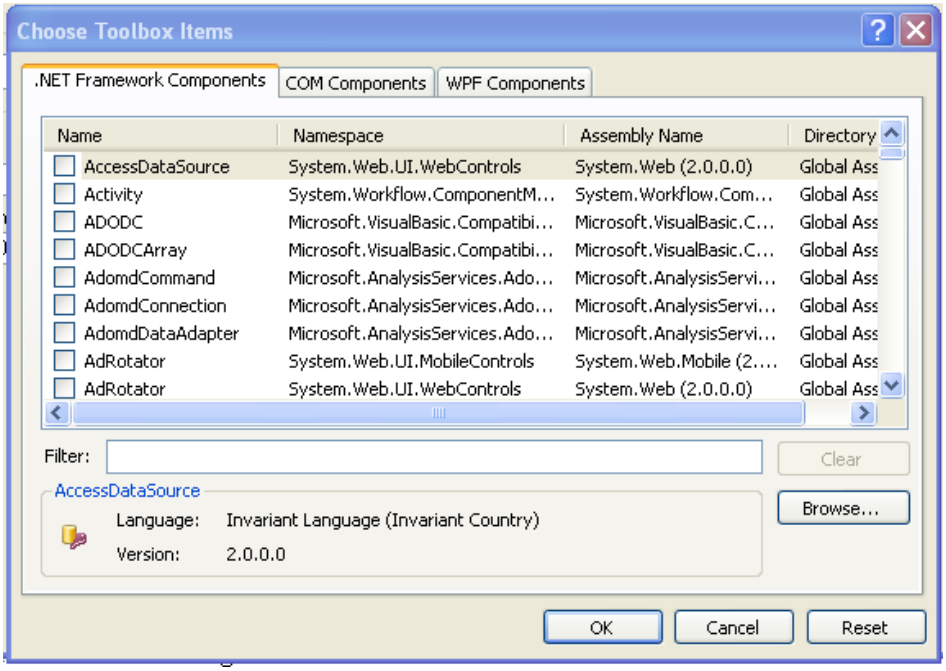
The component Framework is compiled together with the application VTB.
The directory where it is saved, it is the same as the VTB project, The name is defined in Options -> ***Component Name Automatically is added the extension .DLL***

8. Using a NGS Framework component in Visual studio

To use the component you must insert it in a Visual Studio (C#, VBnet or C + +).



Click with the right mouse button on the toolbox of visual studio and when PopUp window appears select the command Choose Items



In the next window, select the Browser button and then search the component created (ObjectName.dll example **TestFramework.dll**) and insert it into the toolbox by pressing the OK button

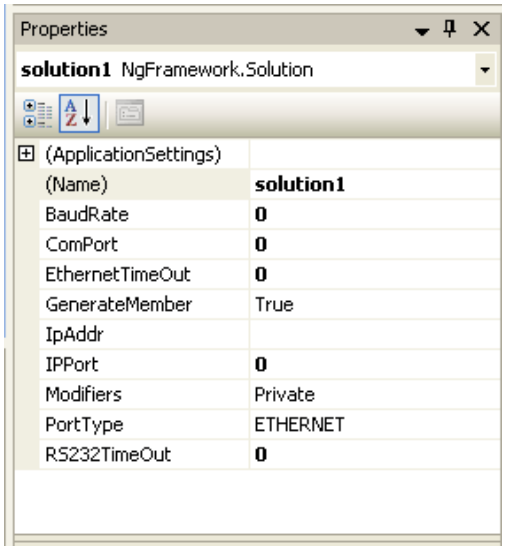
Now you can insert the component in the Visual Studio Project
The DLL component are automatically update when the VTB project is compiled.

WARNING

When you modified the VTB project, the Component is automatically update. You must upload the VTB application in the Hardware platform. If this is not done, the addresses are not updated dellle functions and variables exported by the framework

9. Component Property

The following describes the properties of the component and their meaning. These can be inserted from environment, in the Properties window, or all inside of the code. All of the examples, take as a reference Object Name ***"Solution"***.



Name	Object Name
BaudRate	**Int32** BaudRate for RS232 link (ex: 115200)
ComPort	**Int32** PC Com Port for link to NGS platform
EthernetTimeOut	**Int32** Ethernet Timeout (Ms)
IpAddr	**String** IP ADDR for the NGS hardware The default IP is 10.0.0.80 You can change the IP ADDR by VTB code
IPPort	**Int32** IP PORT(default 6000)
PortType	NgFramework.Solution.TpcCon Type of connection RS32 or ETHERNET. It depends on the hardware used and on the VTB application **"Link Rpc"** The Ethernet link is automatically enabled. This is avaiable only with board ETHERNET PORT
RS232TimeOut	**Int32** RS232 Timeout (Ms)
NumBlockOnEvent	**Int32** Number of blocks needed to generate the event OnTxBlock when using the method PuthEthBlock

10. Set Property by code - recommended system

The following we see how you can set the properties from C # code, Vb etc.
This is recommended because it is possible to read the properties from a configuration file

Set the Ethernet communication - C# code

```
solution1.PortType = NgFramework.Solution.TpcCon.ETHERNET;
solution1.IpAddr = "10.0.0.80";     // IP ADDR
solution1.IPPort = 6000;            // IP PORT (not change this)
solution1.EthernetTimeOut = 5000;   // 5 second time out
```

Set the RS232 communication - C# code

```
solution1.PortType = NgFramework.Solution.TpcCon.RS232;
solution1.ComPort = 1;              // com port  PC
solution1.RS232TimeOut = 1000;      // 1 sec timeout
solution1.BaudRate = 115200;        // baud rate
```

11. Component methods

Standard methods on the component.

Connect

Start the connection Ethernet or RS232.
This method must be called first to started the communication. Before you must set the properties.

Parameter: **None**
Return **None**
Exception **Port type** - Not available
SocketException - Ethernet connection error

Example:

```
solution1.PortType = NgFramework.Solution.TpcCon.ETHERNET;
solution1.IpAddr = "10.0.0.80";           //  IP
solution1.IPPort = 6000;                  // port
solution1.EthernetTimeOut = 5000;         // 5 second
try
{
    solution1.Connect();      // connection start
    // connection ok
}
catch (Exception Se)
{
    // connection failed
      MessageBox.Show(Se.Message, "ERROR !!!",
      MessageBoxButtons.OK,MessageBoxIcon.Error);
}
```

Close

Closes the Connection
This method must be called at end activity

Parameter: **None**
Return ***None***
Exception **No Connect** - No connection open

Example:

```
solution1.Close();
```

CallIfs

Not Use

OpendeviceFlash

Not Use

ClosedeviceFlash

Not Use

WriteFlash

Not use

ReadByte

Reads a Byte from NG memory.
This method is simplified by possibility to variable export

Parameter: **Int32 Addr** NG memory address
Return **Byte** Read value
Exception **SocketException** Connection error

Example:

```
Byte Val;
Val=solution1.ReadByte(Addr);
```

ReadCharMemory

Reads an array of Byte from NG memory
This method is simplified by possibility to variable export

Parameter: **Int32 Addr** Address start
Int32 Len data len
Return **Byte[]** Value
Exception **SocketException** Connection error

Example:

```
Byte[] Val;
Val=solution1.ReadCharMemory(Addr,10);    // Read 10 bytes
```

ReadDouble

Reads a Double value from NG memory.
This method is simplified by possibility to variable export

Parameter: **Int32 Addr** NG memory address
Return ***Double*** Read value
Exception **SocketException** Connection error

Example:

```
Double Val;
Val=solution1.ReadDouble(Addr);
```

ReadInt32Memory

Reads an Int32 array from NG memory.
This method is simplified by possibility to variable export

Parameter: **Int32 Addr** Address start
Int32 Len data len
Return ***Int32*[]** Value
Exception **SocketException** Connection error

Example:

```
Int32[] Val;
Val=solution1.ReadInt32Memory(Addr,10);   // Read 10 Int32
```

ReadInt16

Reads an Int16 value from NG memory.
This method is simplified by possibility to variable export

Parameter: **Int32 Addr** NG memory address
Return ***Int16*** Read value
Exception **SocketException** Connection error

Example:

```
Int16 Val;
Val=solution1.ReadInt16(Addr);
```

ReadInt32

Reads an Int32 value by NG memory.
This method is simplified by possibility to variable export

Parameter: **Int32 Addr** NG memory address
Return ***Int32***Read value
Exception **SocketException** Connection error

Example:

```
Int32 Val;
Val=solution1.ReadInt32(Addr);
```

ReadIntMemory

Reads an Int16 Array from NG Memory
This method is simplified by possibility to variable export

Parameter: **Int32 Addr** Address start
Int32 Len data len
Return ***Int16*[]** Value
Exception **SocketException** Connection error

Example:

```
Int16[] Val;
Val=solution1.ReadIntMemory(Addr,10);     // Read 10 Int16
```

ReadSbyte

Reads a Sbyte (signed Byte) value from NG memory.
This method is simplified by possibility to variable export

Parameter: **Int32 Addr** NG memory address
Return ***Sbyte*** Read value
Exception **SocketException** Connection error

Example:

```
Sbyte Val;
Val=solution1.ReadSbyte(Addr);
```

ReadUint16

Reads an Uint16 (unsigned int) value from NG memory.
This method is simplified by possibility to variable export

Parameter:	**Int32 Addr** NG memory address
Return	***Uint16*** Read value
Exception	**SocketException** Connection error

Example:

```
Uint16 Val;
Val=solution1.ReadUint16(Addr);
```

WriteByte

Writes a Byte value in NG memory.
This method is simplified by possibility to variable export

Parameter:	**Int32 Addr** NG memory address
	Byte Value to write
Return	***None***
Exception	**SocketException** Connection error

Example:

```
Byte Val;
Val=120
solution1.WriteByte(Addr,Val);
```

WriteCharMemory

Writes a Byte array in NG memory.
This method is simplified by possibility to variable export

Parameter:	**Int32 Addr** NG memory address
	Int32 Len Data len
	Byte[] Array Value to write
Return	***None***
Exception	**SocketException** Connection error

Example:

```
Byte[] Val;
Val=new Byte[20];
for(int n=0;n<20;n++)
      Val[n]=n;
solution1.WriteCharMemory(Addr,Val.Length,Val);
```

WriteDouble

Writes a Double value in NG memory.
This method is simplified by possibility to variable export

Parameter:	**Int32 Addr** NG memory address
	Double Value to write
Return	***None***
Exception	**SocketException** Connection error

Example:

```
Double Val;
Val=1234.897
solution1.WriteDouble(Addr,Val);
```

WriteInt16

Writes an Int16 value in NG memory.
This method is simplified by possibility to variable export

Parameter:	**Int32 Addr** NG memory address
	Int16 Value to write
Return	***None***
Exception	**SocketException** Connection error

Example:

```
Int16 Val;
Val=18000;
solution1.WriteInt16(Addr,Val);
```

WriteInt32

Writes an Int32 value in NG memory.
This method is simplified by possibility to variable export

Parameter:	**Int32 Addr** NG memory address
	Int32 Value to write
Return	***None***
Exception	**SocketException** Connection error

Example:

```
Int32 Val;
Val=180000;
solution1.WriteInt32(Addr,Val);
```

WriteInt32Memory

Writes an Int32 array in NG memory.
This method is simplified by possibility to variable export

Parameter:	**Int32 Addr** NG memory address
	Int32 Len Data len
	Int32[] Array Value to write
Return	***None***
Exception	**SocketException** Connection error

Example:

```
Int32[] Val;
Val=new Byte[20];
for(int n=0;n<20;n++)
      Val[n]=n;
solution1.WriteInt32Memory(Addr,Val.Length,Val);
```

WriteIntMemory

Writes an Int16 array in NG memory.
This method is simplified by possibility to variable export

Parameter:	**Int32 Addr** NG memory address
	Int32 Len Data len
	Int16[] Array Value to write
Return	***None***
Exception	**SocketException** Connection error

Example:

```
Int16[] Val;
Val=new Byte[20];
for(int n=0;n<20;n++)
      Val[n]=n;
solution1.WriteInt16Memory(Addr,Val.Length,Val);
```

WriteSbyte

Writes a Sbyte value in NG memory.
This method is simplified by possibility to variable export

Parameter:	**Int32 Addr** NG memory address
	Sbyte Value to write
Return	***None***
Exception	**SocketException** Connection error

Example:

```
Sbyte Val;
Val=120
solution1.WriteSbyte(Addr,Val);
```

WriteUint16

Writes an Uint16 (unsigned int) value in NG memory.
This method is simplified by possibility to variable export

Parameter:	**Int32 Addr** NG memory address
	Uint16 Value to write
Return	***None***
Exception	**SocketException** Connection error

Example:

```
Uint16 Val;
Val=1200
solution1.WriteUint16(Addr,Val);
```

PuthEthBlock

Writes a data block in NG memory.
This method is available only in ETHERNET.
This is used for quickl the data write

Parameter:	**Int32 AddrBase** Start Addr to NG
	Data Data array (Type Int32 – Int16 Byte)
	Int32 LenData Len data
Return	***None***
Exception	**SocketException** Connection error

3 Overload

Example:

```
Int32[] Dati = new Int32[20000];
for (int n = 0; n < 20000; n++)
      Dati[n] = n;
solution1.PuthEthBlock(AddrBase, Dati, 20000);
```

12. Examples of use

You must init the connection before using the examples

Set the Ethernet communication - C# code

```
solution1.PortType = NgFramework.Solution.TpcCon.ETHERNET;
solution1.IpAddr = "10.0.0.80";      // IP ADDR
solution1.IPPort = 6000;             // IP PORT (not change this)
solution1.EthernetTimeOut = 5000;          // 5 second time out
```

Set the RS232 communication - C# code

```
solution1.PortType = NgFramework.Solution.TpcCon.RS232;
solution1.ComPort = 1;                     // com port  PC
solution1.RS232TimeOut = 1000;   // 1 sec timeout
solution1.BaudRate = 115200;               // baud rate
```

Read/Write Int32 Variable

VTB variable export TESTVAR
Type Long (int32 in .NET)
Class CsProva

```
Int32 Val;
solution1.CsProva.TESTVAR = 100;    // Write value 100
Val = solution1.CsProva.TESTVAR;    // Read the value
```

Read Variable Address

```
Int32 Val;
Addr = solution1.CsProva.GetAddrTESTVAR();// Return Int312 Address variable
```

Read/Write Int32 Array

VTB Array Export TESTVAR1(20)
Type Long (int32 in .NET)
Class CsProva

```
Int32 Val;
solution1.CsProva.TESTVAR1[5] = 20;      // Write value 20 to index 5
Val = solution1.CsProva.TESTVAR1[5];     // Read Index 5
```

Read/Write Structure

Vtb Structure Export PSSTRU1
Members Long PAR1
Int PAR2
Char PAR3
Class PSSTRU1 (this class is can not be changed)

```
solution1.PSSTRU1.PAR1 = 100;
solution1.PSSTRU1.PAR2 = 1000;
solution1.PSSTRU1.PAR3 = 20;
```

Read/Write Structure Array

Vtb Structure Export	PSSTRU(15)
Members	Long PAR1
	Char PAR(10)
	Int PAR3
	Long PAR4(5)
Class	PSSTRU (this class is can not be changed)

```
solution1.PSSTRU[1].PAR1 = 120000;
solution1.PSSTRU[1].PAR[3] = 200;
solution1.PSSTRU[1].PAR3 = 20;
solution1.PSSTRU[1].PAR4[2] = 1300;
```

Call a VTB function

VTB Function declaration (Main Page section Page Functions)

```
'*******************************
' Function provaf Declaration
'*******************************
Function provaf(par1 as long) as long $_EXPORT_$ TESTFUNZ
      testvar2=par1
      provaf=1 'return value
endfunction

'*******************************
' Function provaf1 Declaration
'*******************************
function provaf1() as long $_EXPORT_$ TESTFUNZ
      provaf1=testvar2 'return value
endfunction

'*******************************
' Function provaf2 Declaration
'*******************************
function provaf2() as void $_EXPORT_$ TESTFUNZ1
      testvar2=1000
endfunction
```

Use in the .NET

```
Int32 V=solution1.TestFunz.PROVAF(8000);
Int32 V1 = solution1.TestFunz.PROVAF1();
solution1.TestFunz1.PROVAF2();
```

Index