

NGS



# NGS FRAMEWORK

Muoversi in Armonia



*Questo manuale l' applicativo FrameWork sono concessi su licenza e possono essere utilizzati solo secondo i termini della licenza stessa. Le informazioni contenute nel manuale sono solo a scopo informativo e possono subire variazioni senza preavviso e non devono essere intese con alcun impegno da parte di **Promax srl**. **Promax srl** non si assume nessuna responsabilità o obblighi per errori o imprecisioni che possono essere riscontrate in questo manuale. Eccetto quanto concesso dalla licenza, nessuna parte di questa pubblicazione può essere riprodotta, memorizzata in un sistema di archiviazione o trasmessa in qualsiasi forma o con qualsiasi mezzo, elettronico, meccanico, di registrazione o altrimenti senza previa autorizzazione di Promax srl. Qualsiasi riferimento a nomi di società e loro prodotti è a scopo puramente dimostrativo e non allude ad alcuna organizzazione reale.*

**Promax s.r.l.**

**Via Newton, 5G - CastelFiorentino (FI) ITALY [www.promax.it](http://www.promax.it) [info@promax.it](mailto:info@promax.it)**

## DESCRIZIONE GENERALE

NGS Framework è un componente creato da VTB che viene utilizzato su sistemi di sviluppo .NET e compatibili. Il componente mette a disposizione una serie di proprietà e metodi che permettono un interfacciamento semplice e intuitivo per le piattaforme Hardware NG35,NGM13 e compatibili.

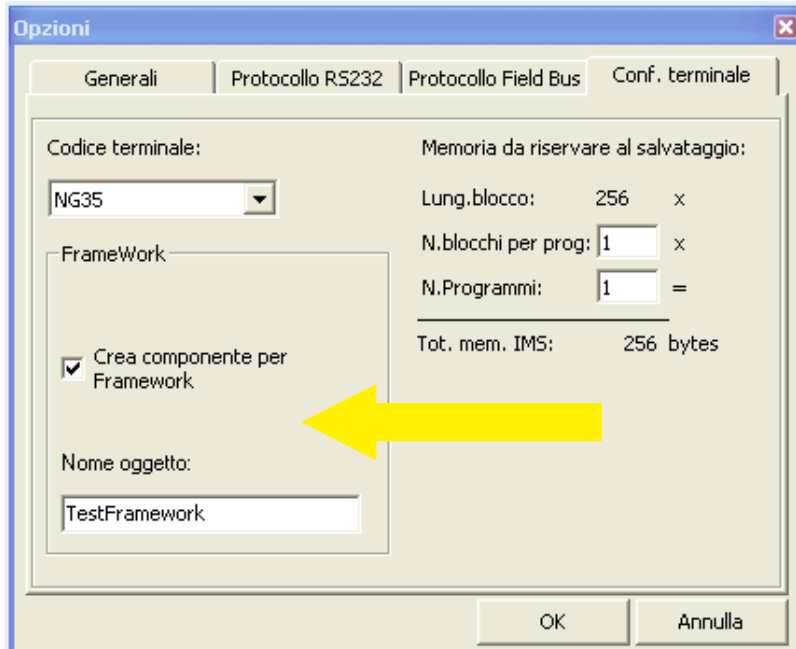
In sostanza utilizzando l' ambiente di sviluppo Microsoft Visual Studio 2005 o successivi, si possono creare applicazioni Windows XP o Windows CE che integrano le risorse di un sistema Promax NGS.

Il risultato è quello di ottenere la flessibilità dei prodotti .NET unita alla potenza dei sistemi NGS.

## Creazione di un componente framework

Per creare un componente da utilizzare in ambienti Visual Studio è necessario configurare VTB per abilitare la creazione di componenti per framework.

Questo viene effettuato dal Menu Opzioni (La creazione del componente Framework è abilitata solamente se viene configurato come tipo di terminale NG35 o NGM13).



Crea componente per Framework

Abilitare questa opzione per creare il componente

Nome oggetto:

TestFramework

Inserire un nome che definisce il tipo di oggetto che viene caricato nell'ambiente visual studio (in questo caso TestFramework.DLL)

Successivamente occorre decidere le variabili e le funzioni da esportare nel componente framework. Esistono i seguenti vincoli:



**Le variabili devono essere globali**

**Non è possibile esportare variabili Bit ma solamente la loro variabile sorgente**

**E' possibile solamente esportare variabili interne,static e fixed**

**Possano essere esportate solamente funzioni della pagina MAIN (globali)**

### Esportazione di una variabile

Nel campo di dichiarazione VTB, occorre spuntare la casella EXP e inserire eventualmente il nome della classe in cui si desidera esportare la variabile. Non è obbligatorio inserire il nome della classe, ma questo semplifica il raggruppamento delle variabili organizzandole per tipologia, campo di utilizzo ecc. Solamente le strutture, vengono esportate in una classe che prende il nome della variabile dichiarata in VTB.

VAR Interne	VAR Bit	Define	VAR Static	VAR VSD	VAR Fixed
Var1		LONG	No	EXP <input checked="" type="checkbox"/>	ClassTest
Variabile	Tipo	Condivisa	Esporta in classe		

Abilitazione ad esportazione della variabile

Nome della classe in cui viene esportata

### Esportazione di una funzione

Per esportare una funzione è stato aggiunto in VTB un ulteriore parametro non obbligatorio durante la fase di dichiarazione della funzione:

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

**\$\_EXPORT\_\$** Parola chiave che identifica l' esportazione della funzione nel framework  
**ClassFTest** Classe di esportazione della funzione (campo non obbligatorio, se omesso la funzione viene esportata nella classe Generic, altrimenti nella classe indicata)



**Occorre sempre dare un nome diverso per le classi di esportazione delle variabili e quello delle funzioni. Pertanto rispettare come nell' esempio sopra, le variabili è stata esportata nella classe ClassTest e la funzione in una classe diversa ClassFTest.**

**Ovviamente è possibile esportare più variabili o funzioni nella medesima classe**

### Esportazione delle variabili **\_SYSTEM**

Tutte le variabili **\_System** vengono esportate automaticamente nella classe **SystemVar**.

### Esportazione delle funzioni di sistema

Tutte le le funzioni di sistema sono raccolte nelle seguenti classi:

#### **BoardNg**.ClasseSpecifica

Dove ClasseSpecifica:

##### **CanOpen**

Contiene tutte le funzioni relative al canopen

##### **Hardware**

Contiene tutte le funzioni relative all' hardware (gestione risorse)

##### **Interpola**

Contiene tutte le funzioni relative all' interpolazione assi

Questa classe è definita solo per la scheda NG35.

Per quanto riguarda la scheda NGM13, le funzioni di interpolazione non sono definite dal sistema, ma dall' applicazione utente, pertanto verranno decise dall' operatore in quale classe e con quale nome vengono esportate.



**Tutte le funzioni di sistema sono contenute in file che può essere aggiornato dinamicamente da REALUPDATE. Pertanto queste possono variare come numero e come ClasseSpecifica.**

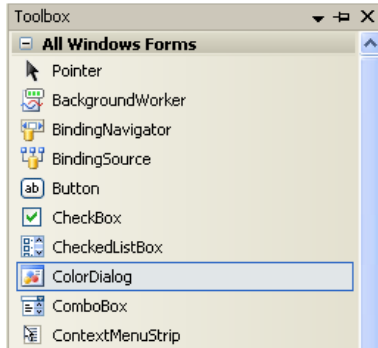
### Compilazione del Framework

Il componente Framework viene compilato insieme all' applicazione VTB.

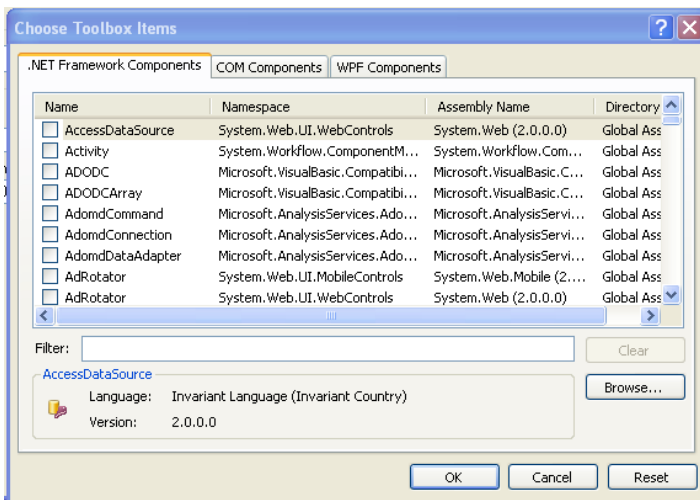
La directory dove questo verrà salvato, sarà la stessa del progetto VTB, mentre il suo nome sarà quello indicato nell' opzioni alla voce **Nome Oggetto** con estensione **.DLL**.

## Inserimento del componente frame work in un progetto di VisualStudio

Una volta creato il componente è sufficiente inserirlo in un progetto di Visual Studio (c#, VB o C++) per poterlo utilizzare.



Cliccare con il tasto Destro sulla toolbox di visual studio e quando appare la finestra PopUp selezionare il comando Choose Items (per inglese) oppure Scegli Elementi (per italiano)



Nella finestra accanto, selezionare il tasto Browser e successivamente cercare il componente create (Nome Oggetto.dll) e inserirlo nella toolbox premendo il pulsante Ok della finestra accanto).



A questo punto il componente si trova nella toolbox e può essere inserito nel progetto

## PROPRIETÀ DEL COMPONENTE FRAMEWORK

Di seguito vengono descritte le proprietà del componente e il loro significato. Queste possono essere inserite dall' ambiente, nella finestra proprietà, oppure all' interno del codice. Tutti gli esempi riportati, prendono come riferimento un Nome Oggetto **“Solution”**.

### Elenco proprietà visibile da ambiente di sviluppo

(ApplicationSettings)	
(Name)	<b>solution1</b>
BaudRate	<b>0</b>
ComPort	<b>0</b>
EthernetTimeOut	<b>0</b>
GenerateMember	True
IpAddr	
IPPort	<b>0</b>
Modifiers	Private
PortType	ETHERNET
RS232TimeOut	<b>0</b>

<b>Name</b>	Nome dell' oggetto
<b>BaudRate</b>	Int32 BaudRate per comunicazione RS232 (es: 115200)
<b>ComPort</b>	Int32 Porta di comunicazione di LINK del PC (es 1=COM1)
<b>EthernetTimeOut</b>	Int32 Timeout in Ms per risposte in Ethernet
<b>IpAddr</b>	String Indirizzo IP della scheda NG Ethernet (es 10.0.0.80)
<b>IPPort</b>	Int32 Porta della scheda NG (default 6000)
<b>PortType</b>	NgFramework.Solution.TpcCon tipo di porta Ethernet o RS232
<b>RS232TimeOut</b>	Int32 Timeout in Ms per risposte in RS232
<b>NumBlockOnEvent</b>	Numero di blocchi necessari per generare l' evento OnTxBlock quando si usa il metodo PuthEthBlock

### Impostazioni proprietà da codice

Di seguito vediamo come è possibile impostare le proprietà dal codice C#,Vnb ecc..

Questa operazione è consigliabile in quanto è possibile leggere le proprietà da un file di configurazione

#### Impostazione comunicazione in Ethernet (solo NG35)

```
solution1.PortType = NgFramework.Solution.TpcCon.ETHERNET;
solution1.IpAddr = "10.0.0.80"; // indirizzo IP
solution1.IPPort = 6000; // porta di connessione
solution1.EthernetTimeOut = 5000; // 5 secondi
```

#### Impostazione comunicazione in RS232

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

### Metodi standard del componente

Il componente presenta sempre dei metodi standard a prescindere che vengono esportate delle variabili o funzioni.

**Connect**

Avvia la connessione in Ethernet o RS232.

Questo metodo deve essere chiamato prima di iniziare le attività di connessione, ma dopo avere impostato le proprietà della connessione.

<b>Parametri:</b>	Nessuno	
<b>Valore di ritorno</b>	Nessuno	
<b>Eccezioni</b>	No Type	Tipo di porta non disponibile
	SocketException	Errore relativo alla connessione

**Esempio:**

```

solution1.PortType = NgFramework.Solution.TpcCon.ETHERNET;
solution1.IpAddr = "10.0.0.80"; // indirizzo IP
solution1.IPPort = 6000; // porta di connessione
solution1.EthernetTimeout = 5000; // 5 secondi
try
{
    solution1.Connect(); // apre la connessione
    // connection ok
}
catch (Exception Se)
{
    // connection failed
    MessageBox.Show(Se.Message, "ERRORE", Doc. N. PX020608
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}

```

NGS FrameWork

Rev. 1.0 - 17/02/2009

Doc. N. PX020608

**Close**

Chiude la connessione precedentemente aperta

Questo metodo deve essere chiamato alla fine delle attività di connessione

<b>Parametri:</b>	Nessuno	
<b>Valore di ritorno</b>	Nessuno	
<b>Eccezioni</b>	No Connect	nessuna connessione aperta

**Esempio:**

```

solution1.Close();

```

**Callfs**

Primitiva di chiamata ad una funzione interna NG

Metodo Obsoleto se vengono esportate le funzioni (operazione consigliata).

Non è consigliabile chiamare questo metodo, in quanto Ngsframework mette a disposizione la possibilità i esportare le funzioni

**OpencodeFlash**

Apertura device Flash scheda NG (solo NG35)

Il metodo deve essere chiamato prima di scrivere in flash

<b>Parametri:</b>	Int32 Addr	Indirizzo di scrittura in flash
<b>Valore di ritorno</b>	Nessuno	
<b>Eccezioni</b>	DeviceNotClosed	Il device è già stato aperto
	SocketException	Errore relativo alla connessione

**Esempio:**

```

solution1.OpenDeviceFlash(Addr);

```

**CloseDeviceFlash** Chiude il device Flash scheda NG (solo NG35)

Il metodo deve essere chiamato dopo la scrittura in flash

<b>Parametri:</b>	nessun	
<b>Valore di ritorno</b>	Nessuno	
<b>Eccezioni</b>	DeviceNotOpen	Nessun device aperto
	SocketException	Errore relativo alla connessione

*Esempio:*

```
solution1.CloseDeviceFlash();
```

---

**WriteFlash** Scrive una array di Byte nella Flash della scheda NG (solo Ng 35) partendo dall' indirizzo selezionato in OpenDeviceFlash

<b>Parametri:</b>	Int32 Len,Byte[] Array	
<b>Valore di ritorno</b>	Nessuno	
<b>Eccezioni</b>	Device Not Open	Device non aperto
	SocketException	Errore relativo alla connessione

*Esempio:*

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

---

**ReadByte** Legge un byte dalla memoria NG. Tale metodo viene reso più immediato dalla possibilità di esportare le variabili.

<b>Parametri:</b>	Int32 Addr Indirizzo memoria	
<b>Valore di ritorno</b>	Byte Valore letto	
<b>Eccezioni</b>	SocketException	Errore relativo alla connessione

*Esempio:*

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

---

**ReadCharMemory** Legge un array di byte dalla memoria NG. Tale metodo viene reso più immediato dalla possibilità di esportare le variabili.

<b>Parametri:</b>	Int32 Addr Indirizzo memoria, Int32 Len dati	
<b>Valore di ritorno</b>	Byte[] Valori letti	
<b>Eccezioni</b>	SocketException	Errore relativo alla connessione

*Esempio:*

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

**ReadDouble** Legge un double dalla memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria  
**Valore di ritorno** Double Valore letto  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

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

**ReadInt32Memory** Legge un array di Int32 dalla memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria, Len dati  
**Valore di ritorno** Int32[] Valori letti  
**Eccezioni** SocketException Errore relativo alla connessione

NGS FrameWork

Rev. 1.0 – 12/06/2008

Doc. N. PX020608

*Esempio:*

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

**ReadInt16** Legge un Int16 dalla memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria  
**Valore di ritorno** Int16 Valore letto  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

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

**ReadInt32** Legge un Int32 dalla memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria  
**Valore di ritorno** Int32 Valore letto  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

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

**ReadIntMemory** Legge un array di Int16 dalla memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria, Len dati  
**Valore di ritorno** Int16[] Valori letti  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

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

**ReadSbyte** Legge uno Sbyte dalla memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria  
**Valore di ritorno** SByte Valore letto  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

```
SByte Val;  
Val=solution1.ReadSByte (Addr) ;
```

**ReadUInt16** Legge uno UInt16 dalla memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria  
**Valore di ritorno** UInt16 Valore letto  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

```
UInt16 Val;  
Val=solution1.ReadUInt16 (Addr) ;
```

**WriteByte** Scrive un Byte nella memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria,Byte Valore  
**Valore di ritorno** Nessuno  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

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

**WriteCharMemory** Scrive un Array di Byte nella memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria,Int32 LenData,Byte[] Array  
**Valore di ritorno** Nessuno  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

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

**WriteDouble** Scrive un Double nella memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria, Double Valore  
**Valore di ritorno** Nessuno  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

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

**WriteInt16** Scrive un Int16 nella memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria, Int16 Valore  
**Valore di ritorno** Nessuno  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

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

**WriteInt32** Scrive un Int32 nella memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria, Int32 Valore  
**Valore di ritorno** Nessuno  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

```
Int32 Val;  
Val=18000;  
solution1.WriteInt16 (Addr, Val);
```

**WriteInt32Memory** Scrive un Array di Int32 nella memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria, Int32 Len, Int32[] Valori  
**Valore di ritorno** Nessuno  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

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

**WriteIntMemory** Scrive un Array di Int16 nella memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria, Int32 Len, Int16[] Valori  
**Valore di ritorno** Nessuno  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

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

**WriteSbyte** Scrive uno SByte nella memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria, Sbyte Valore  
**Valore di ritorno** Nessuno  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

```
SByte Val;
Val=18000;
solution1.WriteSbyte (Addr, Val) ;
```

**WriteUInt16** Scrive un UInt16 nella memoria NG. Tale metodo viene reso piú immediato dalla possibilità di esportare le variabili.

**Parametri:** Int32 Addr Indirizzo memoria, UInt16 Valore  
**Valore di ritorno** Nessuno  
**Eccezioni** SocketException Errore relativo alla connessione

*Esempio:*

```
UInt16 Val;
Val=18000;
solution1.WriteUInt16 (Addr, Val) ;
```

**PuthEthBlock** Scrive un blocco di dati nella scheda NG. Metodo utilizzabile solo in Ethernet. Velocizza l'invio di dati

**Parametri:** **Int32 AddrBase** Indirizzo memoria NG  
**Dati** Vettore di dati da trasferire (tipi gestiti t32, int26, Byte)  
**LenData** Numero di dati da trasferire  
**Valore di ritorno** Nessuno  
**Eccezioni** SocketException Errore relativo alla connessione

3 Overload

*Esempio:*

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

## Esempi di utilizzo del componente framework

### Esempio Lettura e scrittura di una variabile INT32

*Variabile esportata da VTB*      TESTVAR

*Tipo*                              Long (int32)

*Classe*                            CsProva

```
Int32 Val;
solution1.CsProva.TESTVAR = 100;      // scrive il valore 100
Val = solution1.CsProva.TESTVAR;      // legge la variabile
```

### Letture indirizzo tramite il metodo GetAddr

```
Addr = solution1.CsProva.GetAddrTESTVAR(); // Ritorna un Int32 addr variabile
dell scheda NG
```

### Esempio Lettura e scrittura di un array INT32

*Array esportato da VTB*          TESTVAR1(20)

*Tipo*                              Long (int32)

*Classe*                            CsProva

```
Int32 Val;
solution1.CsProva.TESTVAR1[5] = 20;      // scrive il valore 20 all' indice 5
Val = solution1.CsProva.TESTVAR1[5];      // legge la variabile dall' indice 5
```

### Letture indirizzo tramite il metodo GetAddr

```
Addr = Val = solution1.CsProva.GetAddrTESTVAR1(); // Ritorna un Int32 addr
variabile dell scheda NG
```

### Esempio di una struttura

*Struttura esportata da VTB*      PSSTRU1

*Membri della struttura*          Long PAR1

Int PAR2

Char PAR3

*Classe*                            PSSTRU1

```
solution1.PSSTRU1.PAR1 = 100;
```

```
solution1.PSSTRU1.PAR2 = 1000;
```

```
solution1.PSSTRU1.PAR3 = 20;
```

### Esempio di un array di strutture

*Struttura esportata da VTB* PSSTRU(15)  
*Membri della struttura* Long PAR1  
Char PAR(10)  
Int PAR3  
Long PAR4(5)  
*Classe* PSSTRU

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

### Esempio di gestione di una variabile SYSTEM

```
Int32 V=solution1.SystemVar._SYSTEM_PXC; // Legge la quota asse X interpolatore
```

### Esempio di chiamata funzioni remote esportate

*Dichiarazione in VTB (pagina MAIN funzioni di pagina)*

```
Function provaf(par1 as long) as long $_EXPORT_$ TESTFUNZ  
testvar2=par1  
provaf=1  
endfunction  
function provaf1() as long $_EXPORT_$ TESTFUNZ  
provaf1=testvar2  
endfunction  
function provaf2() as void $_EXPORT_$ TESTFUNZ1  
testvar2=1000  
endfunction
```

### Utilizzo nel framework

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

**Esempio di chiamata funzione di sistema**

```
// interpolazione lineare su 6 assi
Byte Ret = solution1.BoardNg.Interpola.Px_Moveto(1000, 1, 2000, 3000, 4000,
5000, 6000, 7000);
solution1.BoardNg.Hardware.Ng_Dac(1, 1000); // write canale analogico 1 su NG35
UInt16 Val=solution1.BoardNg.Hardware.Ng_Adc(1); // legge canale analogico 1
```