

ISO Ns – Next Step

Utilizzo del componente per .NET



Rev. 1.2.2 © Promax srl



1 PREFAZIONE

In questo manuale viene descritto l' utilizzo del componente **NsWork.DLL** per ambienti .NET.

Tale componente corrisponde agli standard di utilizzo e pertanto vengono spiegate solo le funzionalità dei metodi ed eventi del componente stesso partendo dal presupposto che l' utilizzatore conosca gli ambienti di sviluppo per il framework NET.

1.1 Prima di iniziare

L' utilizzo di **NsWork.DLL** prevede che esista il file di configurazione "**IsoNs.cfg**" del CN già installato nella stessa cartella. Per il file di configurazione si rimanda al manuale tecnico di IsoNs.

Il file di configurazione contiene tutte le informazioni necessarie al componente per il corretto funzionamento e l' adattamento al CN in uso.

Nella stessa cartella di **NsWork.DLL** devono trovarsi le seguenti DLL:

Compiler.DLL

ComSynk.DLL

Queste vengono direttamente allocate dal componente stesso e pertanto è solamente necessaria la loro presenza.

Il componente viene caricato nell' ambiente di sviluppo come tutti gli altri componenti del framework eseguendo lo standard del loro utilizzo.

Per prima cosa occorre assegnare alla proprietà SetCn l' indirizzo del CN che dobbiamo utilizzare (normalmente 0) registrando le seguenti chiavi:

```
private const String KeyName = "HKEY_CURRENT_USER\\Software\\Promax\\ISONs\\InstallPath";
private const String NomeConf = "HKEY_CURRENT_USER\\Software\\Promax\\ISONs\\NomeConfig";

// lancia applicazione
private void StartIsoNs()
{
    // registra applicaziobe
    Microsoft.Win32.Registry.SetValue(KeyName, "ISONs", Application.StartupPath);
    Microsoft.Win32.Registry.SetValue(NomeConf, "ISONs", "IsoNs.Cfg");
    IsoNs.SetCn(0);
}
```

Proprietà , metodi e eventi di NsWork

Di fatto NsWork non ha nessuna proprietà impostabile dall' ambiente in quanto tutta la sua configurazione iniziale viene presa dal file di configurazione "**IsoNs.cfg**".

2 Eventi Disponibili

2.1 AbsRelChanged

Evento che si verifica al cambiamento dello stato del CN da quote assolute o relative. Generato a seguito di un'istruzione G90 G91 o da funzioni manuali. Un valore false indica che il CN si trova in movimentazione assoluta dallo zero pezzo o zero macchina, viceversa true indica che il CN si trova in movimentazione relativa dallo zero pezzo o zero macchina.

Esempio c#:

```
private void AbsRelChanged(object sender, NsWork.BoolArgs e)
{
    if (e.Value == false)
        ... Movimentazione assoluta
    else
        ... Movimentazione relativa
}
```

2.2 DiamChanged

Evento che si verifica al cambiamento delle impostazioni del diametro utensile. Generato in seguito ad un'istruzione D del partprogram. Il valore e' contenuto in un intero e rappresenta i millesimi di millimetro del diametro utensile.

Esempio c#:

```
private void DiamChanged(object sender, NsWork.GenArgs e)
{
    m_ActDiam = e.Value;
}
```

2.3 DigitalInputChanged

Evento che si verifica al cambiamento dello stato di un ingresso digitale. Gli ingressi digitali abilitati alla generazione degli eventi vanno prima configurati nel file **IsoNs.cfg** nel seguente modo:

READ_INPUT=inputn,inputn1,inputn2 ecc.

Dove inputn,n1,n2 corrispondono al numero dell'ingresso abilitato a generare gli eventi. Es:

READ_INPUT=1,5,32 Ingressi 1-5 e 32 abilitati alla generazione eventi

Esempio c#:

```
private void DigitalInputChanged(object sender, NsWork.DigitalInputArgs e)
{
    for(int n=0;n<e.DigitalInput.Length;n++)
    {
        // e.DigitalInput e' un vettore di interi che contiene gli ingressi cambiati
        // es: e.DigitalInput[0]=1 Ingresso 1 cambiato
        // e.DigitalStato e' un vettore di boolean
        // che contiene lo stato del relativo ingresso
        // e: e.DigitalStato[0]=false Ingresso 1 a false
        // test su ingresso 1
        if(e.DigitalInput[n]==1)
        {
            if(e.DigitalStato[n]==true)
            {
                if(IsoNs.IsStatusRun==true) // se il CN e' in RUN
                    IsoNs.ProgramRun.PauseProg();
            }
        }
    }
}
```

2.4 EnabledChanged

Evento che si verifica al cambiamento dello stato di abilitazione del driver di un asse
Value contiene lo stato di Enable True asse abilitato false asse disabilitato
IndexAxis contiene l' indice dell' asse (0-1-2) es X,Y,Z

Esempio c#:

```
private void EnabledAxisChanged(object sender, NsWork.HEArgs e)
{
    if (e.Value == false)
        e.IndexAxis... // indice asse disabilitato
    else
        e.IndexAxis... // indice asse abilitato
}
```

2.5 ErrorChanged

Evento che si verifica quando il CN trova un errore.
Value contiene lo stato di errore True CN in errore false CN Ok
I codici di allarme vengono configurati nel **IsoNs,cfg**
Gli errori vengono letti dal metodo **CnError.GetError()** di NsWorks.dll
Tale metodo ritorna un array di String che contiene la descrizione dei vari errori

Esempio c#:

```
private void ErrorChanged(object sender, NsWork.ErrorArgs e)
{
    if (e.Value == false)
        ... Ripristino errori
    else
    {
        String[] Err;
        Err=IsoNs .CnError.GetError();
        if(Err!=null)
            Err[n]=descrizione errore
    }
}
```

2.6 ExtRunChanged

Evento che si verifica al cambiamento dell' ingresso di RUN programma esterno.
L' ingresso viene definito sul CN nell' applicativo VTB
Value contiene lo stato True Richiesta di RUN da ingresso esterno

Esempio c#:

```
private void ExtRunChanged(object sender, NsWork.BoolArgs e)
{
    if (e.Value == true)
        IsoNs.ProgramRun.ExucuteProg(LineStart); // -1 da inizio programma ISO
}
```

2.7 ExtStopChanged

Evento che si verifica al cambiamento dell' ingresso di STOP programma esterno.
L' ingresso viene definito sul CN nell' applicativo VTB
Value contiene lo stato True Richiesta di STOP da ingresso esterno

Esempio c#:

```
private void ExtStopChanged(object sender, NsWork.BoolArgs e)
{
    if (e.Value == true)
        IsoNs.ProgramRun.StopProg();
}
```

2.8 ExtPauseChanged

Evento che si verifica al cambiamento dell' ingresso di PAUSA programma esterno.
L' ingresso viene definito sul CN nell' applicativo VTB
Value contiene lo stato True Richiesta di PAUSA da ingresso esterno

Esempio c#:

```
private void ExtPauseChanged(object sender, NsWork.BoolArgs e)
{
    if (e.Value == true)
        IsoNs.ProgramRun.PauseProg();
}
```

2.9 FeedChanged

Evento che si verifica al cambiamento della F impostata
L' evento viene generato da istruzioni F o da funzioni manuali
Value contiene in intero della F
Format contiene un valore string formattato all' unita' di misura impostata

Esempio c#:

```
private void FeedChanged(object sender, NsWork.FeedArgs e)
{
    LedFeed.Level = e.Value;
    LblFeed.Text = e.Format;
}
```

2.10 HomeChanged

Evento che si verifica quando un asse ha terminato la funzione di Homing
Value contiene lo stato di Home True asse con homing effettuato
IndexAxis contiene l' indice dell' asse (0-1-2) es X,Y,Z

Esempio c#:

```
private void HomeChanged(object sender, NsWork.HEArgs e)
{
    if (e.Value == false)
    {
        e.IndexAxis... // indice asse che ha perso l' homing
    }
    else
    {
        e.IndexAxis... // indice asse che ha effettuato l' homing
    }
}
```

2.11 LenUtChanged

Evento che si verifica al cambiamento della lunghezza utensile impostata da part
program
Value contiene il valore in millesimi di millimetro della lunghezza utensile

Esempio c#:

```
private void LenUtChanged(object sender, NsWork.GenArgs e)
{
    m_ActLenUt=e.Value;
}
```

2.12 MexecuteChanged

Evento che si verifica al termine di una funzione M interna al CN
Value true M in corso false M terminata

Esempio c#:

```
private void MexecuteChanged(object sender, NsWork.BoolArgs e)
{
    if(e.Value==true)
        // M START
    else
        // M terminata
}
```

2.13 MoveChanged

Evento che si verifica al cambiamento movimento assi
Value true movimento assi in corso False movimento assi terminato

Esempio c#:

```
private void MoveChanged(object sender, NsWork.BoolArgs e)
{
    if (e.Value == false)
        movimento terminato
    else
        movimento in corso
}
```

2.14 NlineaChanged

Evento che si verifica al cambiamento numero di linea TEORICO in esecuzione
Value contiene un intero del numero di linea TEORICO in esecuzione

Esempio c#:

```
private void NlineaChanged(object sender, NsWork.GenArgs e)
{
    Label1.Text=e.Value.ToString();
}
```

2.15 NlineaRealChanged

Evento che si verifica al cambiamento numero di linea REALE in esecuzione sul CN
Questo numero di linea può differenziarsi da quello TEORICO (evento NlineaChanged) se si lavora con G60 (interpolazione veloce). In effetti viene considerata la PROFONDITA' del BUFFER. Quindi in questo casom il numero di linea REALE è sempre indietro al numero di LINEA TEORICO (la differenza dipende dalla profondità del BUFFER impostata)
Value contiene un intero del numero di linea REALE in esecuzione

Esempio c#:

```
private void NlineaRealChanged(object sender, NsWork.GenArgs e)
{
    Label1.Text=e.Value.ToString();
}
```

2.16 OnCloseCom

Evento che si verifica alla chiusura delle comunicazioni con il CN

2.17 PauseChanged

Evento che si verifica al cambiamento dello stato di pausa del CN
Value false CN in ripresa da pausa – True CN in pausa

Esempio c#:

```
private void PauseChanged(object sender, NsWork.PauseArgs e)
{
    if (e.Value == false)
        // CN in ripresa da pausa
    else
        // CN in PAUSA
}
```

2.18 PianoChanged

Evento che si verifica al cambiamento del piano di lavoro del CN a seguito delle istruzioni **G17 G18 G19G70**

PianoSet contiene un array di string con i nomi dei due assi del piano di lavoro settato

Esempio c#:

```
private void PianoChanged(object sender, NsWork.PianoArgs e)
{
    LblPiano1.Text = e.PianoSet[0];
    LblPiano2.Text = e.PianoSet[1];
}
```

2.19 PotVperChanged

Evento che si verifica al cambiamento dello stato di abilitazione del potenziometro override esterno

Value false potenziometro esterno disabilitato – True abilitato

Esempio c#:

```
private void PotVperChanged(object sender, NsWork.BoolArgs e)
{
    if (e.Value == false)
        .....Disabilitato
    else
        ..... Abilitato
}
```

2.20 QuoteOffsetChanged

Evento che si verifica al cambiamento degli offset assi in seguito ad un istruzione G93

Format contiene una string del valore formattato della quota asse

Index indice dell' asse

Value Intero che contiene il valore dell' offset asse

Esempio c#:

```
private void QuotaOffsetChanged(object sender, NsWork.ZeriArgs e)
{
    labelval.Text = e.Format;
    Int32 Value = e.Value;
    Int32 ResQuote = IsoNs.Config.NsConfig.GetValueParByName("RESQUOTE");
    Double ValZero = e.Value;
    ValZero /= ResQuote;
}
```

2.21 QuoteOffsetTestaChanged

Evento che si verifica al cambiamento degli offset Testa in seguito ad un istruzione T

Format contiene una string del valore formattato della quota asse

Index indice dell' asse

Value Intero che contiene il valore dell' offset asse

Esempio c#:

```
private void QuotaOffsetTestaChanged(object sender, NsWork.ZeriArgs e)
{
    labelval.Text = e.Format;
    Int32 Value = e.Value;
    Int32 ResQuote = IsoNs.Config.NsConfig.GetValueParByName("RESQUOTE");
    Double ValZero = e.Value;
    ValZero /= ResQuote;
}
```

2.22 QuoteRChanged

Evento che si verifica al cambiamento di una quota assi REALE (se abilitata)

FormatRealValue contiene un array string dei valori formattati delle quote assi ASSOLUTE riferite allo zero macchina

FormatSpaceErr contiene un array string dei valori formattati degli errori di spazio degli assi (differenza tra quota teorica e quota reale)

Format contiene un array string dei valori formattati delle quote assi RELATIVE riferite agli Offset,Zeri asse

RealValue contiene un array Int32 dei valori delle quote assi ASSOLUTE riferite allo zero macchina

Value contiene un array Int32 dei valori delle quote assi RELATIVE riferite agli Offset,Zeri asse

Mask Intero gestito a Bit che indica le quote assi cambiate

Bit 0 Primo Asse - Bit 1 Secondo Asse ecc..

I bit non abilitati, non hanno generato l' evento

Esempio c#:

```
private void QuoteRChanged(object sender, NsWork.QuoteArgs e)
{
    Int32 P = 1;
    for (int n = 0; n < e.FormatValue.Length; n++)
    {
        if ((Mask & P) == P) // quota cambiata
        {
            MyQuoteAbs[n].Text = FormatRealValue[n];
            MyQuoteRel[n].Text = FormatValue[n];
        }
        P <<= 1;
    }
}
```

2.23 QuoteTchanged

Evento che si verifica al cambiamento di una quota assi TEORICA

FormatRealValue contiene un array string dei valori formattati delle quote assi ASSOLUTE riferite allo zero macchina

FormatSpaceErr contiene un array string dei valori formattati degli errori di spazio degli assi (differenza tra quota teorica e quota reale)

Format contiene un array string dei valori formattati delle quote assi RELATIVE riferite agli Offset,Zeri asse

RealValue contiene un array Int32 dei valori delle quote assi ASSOLUTE riferite allo zero macchina

Value contiene un array Int32 dei valori delle quote assi RELATIVE riferite agli Offset,Zeri asse

Mask Intero gestito a Bit che indica le quote assi cambiate

Bit 0 Primo Asse - Bit 1 Secondo Asse ecc..

I bit non abilitati, non hanno generato l' evento

Esempio c#:

```
private void QuoteTchanged(object sender, NsWork.QuoteArgs e)
{
    Int32 P = 1;
    for (int n = 0; n < e.FormatValue.Length; n++)
    {
        if ((Mask & P) == P) // quota cambiata
        {
            MyQuoteAbs[n].Text = FormatRealValue[n];
            MyQuoteRel[n].Text = FormatValue[n];
        }
        P <<= 1;
    }
}
```

2.24 QuoteZeriChanged

Evento che si verifica al cambiamento degli Zeri assi in seguito ad un istruzione G94

Format contiene una string del valore formattato della quota asse

Index indice dell' asse

Value Intero che contiene il valore dell' offset asse

Esempio c#:

```
private void QuotaZeriChanged(object sender, NsWork.ZeriArgs e)
{
    labelVal.Text = e.Format;
    Int32 Value = e.Value;
    Int32 ResQuote = IsoNs.Config.NsConfig.GetValueParByName("RESQUOTE");
    Double ValZero = e.Value;
    ValZero /= ResQuote;
}
```

2.25 RemoveMark

Evento richiesto da Applicazione simula che abilita una richiesta a rimuovere il marcatore di linea ISO

Questo evento ha un utilizzo specifico solo se viene abilitata la simulazione con il file Simulation.dll. Questo file puo' richiedere all' interfaccia principale di rimuovere il marcatore di linea in seguito ad un evento ReqMarkLine

2.26 ReqMarkLine

Evento richiesto da Applicazione simula che abilita una richiesta a a marcare una linea nel part program ISO

Questo evento ha un utilizzo specifico solo se viene abilitata la simulazione con il file Simulation.dll. Questo file puo' richiedere all' interfaccia principale di marcare una linea ISO per evidenziarne la provenienza

2.27 RunStopChanged

Evento che si verifica al cambiamento di stato RUN STOP del CN generato da un RUN PROGRAMMA
Value True indica che il CN e' in RUN – False indica che il CN e in STOP

Esempio c#:

```
private void RunStopChanged(object sender, NsWork.BoolArgs e)
{
    ProgInRun = e.Value;
    if (e.Value == false)
    {
        // il CN e' passato da uno stato di RUN a uno di STOP
    }
    else
        // il CN e' passato da uno stato di STOP a uno di RUN
}
```

2.28 ScriptChanged

Evento che si verifica Al RUN/STOP di codici script ISONS cioè' quando viene invocato il metodo **IsoNs.ProgramRun.ExecuteScript**
Value True indica script in esecuzione – False Script terminato

Esempio c#:

```
private void ScriptChanged(object sender, NsWork.BoolArgs e)
{
    if (e.Value == false)
        .. script terminato
    else
        .. script terminato in esecuzione
}
```

2.29 SpeedChanged

Evento che si verifica al cambiamento della **Speed mandrino** in seguito ad un istruzione S da part program
Value intero che contiene il valore impostato da S
Format Stringa che contiene il valore formattato in base alla configurazione
PercValue Double che contiene la percentuale (in base al Valore Max impostato in configurazione)

Esempio c#:

```
private void SpeedChanged(object sender, NsWork.FeedArgs e)
{
    LedSpeed.Level = e.Value;
    LblSpeed.Text = e.Format;
    LblPerc.Text=e.PercValue.ToString();
}
```

2.30 StepModeChanged

Evento che si verifica al cambiamento dello stato del CN dal modo di esecuzione STEP by STEP (una linea alla volta) al modo di esecuzione continuo.
Evento che si verifica in seguito alla chiamata del metodo:

IsoNs.ProgramRun.StepMode

Value False CN in esecuzione normale – True in esecuzione STEP

Esempio c#:

```
private void StepModeChanged(object sender, NsWork.BoolArgs e)
{
    if (e.Value == false)
        BtnStepMode.ImageIndex = 0;
    else
        BtnStepMode.ImageIndex = 1;
}
```

2.31 TabUtChanged

Evento che si verifica al cambiamento del valore della tabella utensile selezionata.
Funzione ISO Tn

Esempio c#:

```
private void TabUtChanged(object sender, NsWork.GenArgs e)
{
    Label1.Text=e.Value.ToString(); // e. Value contiene il valore della tabella utensile selezionata
}
```

2.32 UserChanged

Evento che si verifica al cambiamento di una variabile USER GENERIC configurata sul CN tramite il file IsoNs.cfg. Le variabili USER GENERIC possono essere utilizzate per scambio dati tra CN e PC (poiché queste sono gestibili anche da applicazione VTB sul CN)

Value Array di Int32 che contiene lo stato delle variabili USER GENERIC

Mask Int32 mappato a bit che indica la variabile cambiata

Bit 0 Variabile 1

Bit 1 Variabile 2

Solo quelle con il Bit ad 1 sono effettivamente cambiate di valore

Esempio c#:

```
private void UserChanged(object sender, NsWork.UserGeneric e)
{
    Int32 P=1;
    for(int n=0;n<e.Value.Length;n++) // controlla quale variabile e' cambiata per non utilizzarle tutte
    {
        if((e.Mask & P)==P) // variabile cambiata
            LblUser.Text = e.Value[n].ToString();
        P <<= 1; // shift bit
    }
}
```

2.33 VperChanged

Evento che si verifica al cambiamento del valore Percentuale di Override della velocità FEED degli assi. Il valore massimo dipende dal tipo di CN e dall' ingresso analogico configurato per la lettura dell' override. Il valore massimo dell' ingresso analogico viene comunque inserito nel file IsoNs.cfg

Value Intero che contiene il valore del potenziometro Override

PercValue valore percentuale

Esempio c#:

```
private void VperChanged(object sender, NsWork.VperArgs e)
{
    LedOverride.Level = e.Value;
    LblOw.Text = e.PercValue + " %";
}
```

2.34 AcqExecute

Evento che si verifica al cambiamento del bit della statusword di acq da sensore.

Value Boolean che indica lo stato di acquisizione

True Acq terminata

False Bit resettato per acq in corso

Esempio c#:

```
private void AcqExecute(object sender, Boolean e)
{
    if(e.Value==True)
        // Acq Terminata
    else
        // Acq Iniziata
}
```

2.35 MolVolChanged

Evento che si verifica al cambiamento del valore del moltiplicatore software per il volante elettronico. Questo evento viene generato anche se il moltiplicatore cambia da applicazione VTB tramite selettore.

Esempio c#:

```
private void MolVolChanged(object sender, NsWork.GenArgs e)
{
    Label1.Text=e.Value.ToString(); // e. Value contiene il valore del moltiplicatore
}
```

2.36 AxisJogChanged

Evento che si verifica al cambiamento del valore dell' asse selezionato per JOG. Questo evento viene generato anche se l' asse cambia da applicazione VTB tramite selettore.

e.value contiene il numero di asse selezionato

| | |
|---|--------|
| 0 | Asse X |
| 1 | Asse Y |
| 2 | Asse Z |
| 3 | Asse A |
| 4 | Asse B |
| 5 | Asse C |
| 6 | Asse U |
| 7 | Asse V |
| 8 | Asse W |

Esempio c#:

```
private void AxisJogChanged(object sender, NsWork.GenArgs e)
{
    Label1.Text=e.Value.ToString(); // e. Value contiene l' asse selezionato per JOG
}
```

2.37 StartImport

Evento che si verifica quando viene eseguita l' istruzione IMPORT
e.ImportFile contiene il testo del file importato.
e.NomeFile contiene la path del file

2.38 EndImport

Evento che si verifica quando viene terminato l' ultimo IMPORT effettuato
e.ImportFile contiene NULL
e.NomeFile contiene la path del file che ha terminato l' esecuzione

3 Metodi e proprietà di NsWork

3.1 Void SetCn (Int32 IndexCn)

Inizializza il CN (per gestione multiprocesso)

IndexCn Valore compreso tra 0-7 che indica il CN in USO (normalmente 0)
Prima di chiamare SetCn occorre avere registrato le chiavi che indicano il percorso e la configurazione da caricare
Genera eccezione su errore configurazione

Esempio c#:

```
private const String KeyName = "HKEY_CURRENT_USER\\Software\\Promax\\ISONS\\InstallPath";
private const String NomeConf = "HKEY_CURRENT_USER\\Software\\Promax\\ISONS\\NomeConfig";

// lancia applicazione
private void StartIsoNs()
{
    // registra applicaziobe
    Microsoft.Win32.Registry.SetValue(KeyName, "ISONS", Application.StartupPath);
    Microsoft.Win32.Registry.SetValue(NomeConf, "ISONS", "IsoNs.Cfg");
    IsoNs.SetCn(0);
}
```

3.2 NsWork.IsoNs.ErrorCompiler[] CompileFileIso(String Text)

Overlay

NsWork.IsoNs.ErrorCompiler[] CompileFileIso(String Text,out Int32 TotalLine)

Metodo che compila il codice del PartProgram da testo rendendolo disponibile alla lavorazione e controllando in anteprima eventuali errori di sintassi.

Ritorna un array di tipo **NsWork.IsoNs.ErrorCompiler[]**

.Nlinea Int32 numero di linea che ha generato l' errore
.TipoErr Stringa con descrizione errore
.Token Stringa Token della linea che ha generato l' errore

TotalLine torna il numero di linee compilate

Esempio c#:

```
// compila il file
private void Compila()
{
    IsoNs.IsoNs.ErrorCompiler[] Err;
    TextIso.Text="G1X100Y100Z100\n";
    Err=IsoNs .CompileFileIso(TextIso.Text);
    // controlla errori di compilazione
    if (Err == null) // nessun errore il partprogram puo' essere eseguito
    {
        IsoNs.LoadCode();
        IsoNs.ProgramRun.ExucuteProg(-1);
    }
    else
    {
        // si sono verificati degli errori li visualizza
        for (int n = 0; n < Err.Length; n++)
        {
            LblLinea.Text+=" Linea: "+Err[n].Nlinea;
            LblTipo.Text+=" Tipo: "+Err[n].TipoErr;
            LblToken.Text+=" Token: "+Err[n].Token;
        }
    }
}
```

3.3 **NsWork.IsoNs.ErrorCompiler[] CompileFromFile(String PathFile,out int32 TotalLine)**

Metodo che compila il codice del PartProgram da file salvato su hardDisk rendendolo disponibile alla lavorazione e controllando in anteprima eventuali errori di sintassi.

Ritorna un array di tipo **NsWork.IsoNs.ErrorCompiler[]**

.Nlinea Int32 numero di linea che ha generato l' errore
.TipoErr Stringa con descrizione errore
.Token Stringa Token della linea che ha generato l' errore

TotalLine torna il numero di linee compilate

```
// compila il file da file
public ErrorCompiler[] CompileFromFile(String PathFile,out Int32 TotalLine)
{
    Compiler.CodeIsoNs.Error[] Err;
    ErrorCompiler[] MyErr;
    PrepareNewcomp();
    NewComp.CompileFromFile(PathFile);
    TotalLine = NewComp.TotalCodeLine;
    Err = null;
    Err = NewComp.GetError();
    if (Err == null)
        return null;
    MyErr = new ErrorCompiler[Err.Length];
    for (int n = 0; n < Err.Length; n++)
    {
        MyErr[n].Nlinea = Err[n].Nlinea;
        MyErr[n].TipoErr = Err[n].TipoErr;
        MyErr[n].Token = Err[n].Token;
    }
    return MyErr;
}
```

3.4 **NsWork.IsoNs.ErrorCompiler[] CompileFromBlock(String PathFile,out int32 TotalLine)**

Metodo che compila il codice del PartProgram da file salvato su hardDisk a blocchi rendendolo disponibile alla lavorazione e controllando in anteprima eventuali errori di sintassi.

Ritorna un array di tipo **NsWork.IsoNs.ErrorCompiler[]**

.Nlinea Int32 numero di linea che ha generato l' errore
.TipoErr Stringa con descrizione errore
.Token Stringa Token della linea che ha generato l' errore

TotalLine torna il numero di linee compilate

```
// compila il file a blocchi
public ErrorCompiler[] CompileFromBlock(String PathFile, out Int32 TotalLine)
{
    Compiler.CodeIsoNs.Error[] Err;
    ErrorCompiler[] MyErr;
    PrepareNewcomp();
    NewComp.CompileFromBlock(PathFile);
    TotalLine = NewComp.TotalCodeLine;
    Err = null;
    Err = NewComp.GetError();
    if (Err == null)
        return null;
    MyErr = new ErrorCompiler[Err.Length];
    for (int n = 0; n < Err.Length; n++)
    {
        MyErr[n].Nlinea = Err[n].Nlinea;
        MyErr[n].TipoErr = Err[n].TipoErr;
        MyErr[n].Token = Err[n].Token;
    }
    return MyErr;
}
```

NOTE SUI DIVERSI METODI DI COMPILAZIONE

I tre diversi metodi di compilazione utilizzano diverse tecnologie.

- CompileFileIso** Compila il file da stringa di testo passata in memoria.
Questo comporta un utilizzo rilevante di risorse del PC come allocazione di memoria
Il vantaggio è dovuto ad una maggiore velocità di compilazione del file
Questo metodo è consigliato per PartProgram inferiore a 10 Mb
- CompilleFromFile** Compila prendendo il file da HardDisk.
Questo fa sì che siano utilizzate meno risorse di memoria del PC.
Utilizzando dispositivi di salvataggio non molto veloci in lettura, si possono avere dei rallentamenti sulla compilazione del PartProgram.
Questo metodo è consigliato per PartProgram inferiori a 60 Mb
- CompileFromBlock** Utilizza una compilazione in memoria a blocchi.
Questo metodo minimizza al massimo le risorse del PC.
Consigliato per PartProgram maggiori di 60Mb

3.5 EVENTI DI COMPILAZIONE

Il compilatore di IsoNs può generare degli eventi relativi allo stato di compilazione del file.

Dichiarazione degli eventi:

MyIso componente NsWork.IsoNs

```
MyIso.NewComp.CompilerPerc += new EventHandler<Compiler.PercArgs>(NewComp_CompilerPerc);  
MyIso.NewComp.StartCompiler += new EventHandler<Compiler.PercArgs>(NewComp_StartCompiler);  
MyIso.NewComp.EndCompiler += new EventHandler<Compiler.PercArgs>(NewComp_EndCompiler);
```

```
// evento di percentuale compilazione da 0-100%  
void NewComp_CompilerPerc(object sender, Compiler.PercArgs e)  
{  
    StatoCompile.Value = e.Perc;  
}  
  
// evento start compilazione generato ad inizio compilazione  
void NewComp_StartCompiler(object sender, Compiler.PercArgs e)  
{  
    StatoCompile.Value = 0;  
}  
// end compiler generato a termine della compilazione  
void NewComp_EndCompiler(object sender, Compiler.PercArgs e)  
{  
  
}
```

3.6 Void GoEvent....()

Metodi utilizzati da ComSynk.DLL per generare glie eventi indicati

3.7 Boolean LoadCode()

Metodo utilizzato per caricare il codice precedentemente compilato

Ritorna **False** Nessun codice da caricare o eventuali errori di compilazione
True Ok programma pronto per essere eseguito

Esempio c#:

```
private void RunProg()
{
    if (IsoNs .LoadCode()== true)
        IsoNs.ProgramRun.ExecuteProg(-1);
    else
        ... nessun programma o errori di compilazione
}
```

3.8 Void MarkLine()

Metodi utilizzati per generare l' evento ReqMarkLine

3.9 Void RemoveMarkLine()

Metodi utilizzati per generare l' evento RemoveMark

3.10 Void ForceEventQuote()

Forza l' evento QuoteChanged, ZeriChanged, OffsetChanged

3.11 Void EndSession()

Fine sessione. Chiude tutti i thread in modo controllato

3.12 GetMarker proprietà di tipo List<Compiler.MarkerCs> - Read

Ritorna la lista dei Marker presenti nel PartProgram

| | |
|----------------|---------------------------------|
| AddrVar | Indirizzo del Marker in memoria |
| NomeVar | Nome della variabile Marker |
| DescrizioneVar | Descrizione del Marker |

4 Classi di NsWork

4.1 BreakPoint

Racchiude tutti i metodi e proprietà per inserimento BreakPoint (Punti di interruzione). I BreakPoint permettono di interrompere il programma in un punto desiderato. Questo viene effettuato al fine del debug stesso di un PartProgram Iso particolarmente complesso.

Una volta che l' esecuzione del programma raggiunge il BreakPoint questo viene interrotto andando in PAUSA.

4.1.1 `Int32[] BreakPoint.GetAllBreakPoint()`

Ritorna un array di Int32 che contiene tutti i numeri di linea dei breakpoint inseriti

4.1.2 `Void BreakPoint.InsertBreakPoint(Int32 LineNumber)`

Inserisce un BreakPoint al numero di linea indicato

4.1.3 `Boolean BreakPoint.IsBreakPoint(Int32 LineNumber)`

Ritorna True se il numero di linea interrogato contiene un BreakPoint

4.1.4 `Void BreakPoint.RemoveAllBreakPoint()`

Rimuove tutti i BreakPoint inseriti nel PartProgram

4.1.5 `Void BreakPoint.RemoveBreakPoint(Int32 LineNumber)`

Rimuove il BreakPoint al numero di linea indicato

4.2 ParTabUtHdCs

Racchiude tutti i metodi e proprietà relative alla gestione dell **TESTE UTENSILE** e dei parametri della **TABELLA UTENSILE**.

4.2.1 **SelectTabUt** proprietà di tipo **Int32** - Read/Write

Ritorna o seleziona una **TABELLA** utensile.

Il valore di SET deve essere compreso tra 0 e il numero massimo di **TABELLA** utensili inserite in configurazione. Viene generata un'eccezione se il valore non è in questo intervallo.

Se la lettura torna il valore -1 significa che nessuna **TABELLA** utensile è selezionata.

4.2.2 **SelectHd** proprietà di tipo **Int32** - Read/Write

Ritorna o seleziona una **TESTA** utensile.

Il valore di SET deve essere compreso tra 0 e il numero massimo di **TESTA** utensili inserite in configurazione. Viene generata un'eccezione se il valore non è in questo intervallo.

Se la lettura torna il valore -1 significa che nessuna **TESTA** utensile è selezionata.

4.2.3 **Double GetParTab(Double Index)**

Legge il parametro utensile indicato da Index della tabella utensili selezionata.

Vengono generate le seguenti eccezioni:

Nessuna tabella utensile selezionata
Parametro non disponibile

Ritorna il valore Double del parametro

4.2.4 **Double GetParHd(Double Index)**

Legge il parametro testa indicato da Index della tabella teste selezionata.

Vengono generate le seguenti eccezioni:

Nessuna tabella teste selezionata
Parametro non disponibile

Ritorna il valore Double del parametro

4.2.5 **Void WriteParTab(Double Val,Int32 IndexTab,Int32 IndexPar)**

Scrive il parametro utensile indicato da **IndexPar** della tabella utensili indicata in **IndexTab** con il valore **Val**

Vengono generate le seguenti eccezioni:

Nessuna tabella teste selezionata
Parametro non disponibile

4.2.6 **Void SaveParTab(Int32 CnNumber)**

Salva i parametri della tabella utensile in memoria (scritti con WriteParTab) nella configurazione su disco in modo permanente. CnNumber indica il numero di CN (generalmente 0) a cui si fa riferimento.

I parametri saranno quindi variati in modo permanente.

4.3 CnError

Racchiude tutti i metodi e proprietà per gestire gli errori del CN al seguito di un evento **ErrorChanged**

4.3.1 `String[] CnError.GetError()`

Ritorna un array di string che contiene gli errori del CN

Esempio c#:

```
private void ErrorChanged(object sender, NsWork.ErrorArgs e)
{
    if (e.Value == false)
        ... Ripristino errori
    else
    {
        String[] Err;
        Err=IsoNs.CnError.GetError();
        if(Err!=null)
            Err[n]=descrizione errore
    }
}
```

4.4 CnIO

Racchiude tutti i metodi e proprietà per gestire tutte le risorse Input Output del CN sia digitali che analogiche. Ovviamente la gestione di queste risorse dipende dal tipo di CN utilizzato e dalla sua configurazione hardware.

4.4.1 `Int32 CnIO.ReadAnalogInput(Int32 Ninput)`

Ritorna un `Int32` che contiene il valore del canale analogico letto in `Ninput`.

Ninput contiene un valore da 0 a N dove N sono il numero dei canali analogici in input configurati -1. 0 e' il primo canale.

Il valore `Int32` che ritorna e' compreso da 0 a valore Max del canale analogico (1024 per scheda NG35 – 4096 per scheda NGM13).

4.4.2 `Int32 CnIO.ReadAnalogOut(Int32 Nout)`

Ritorna un `Int32` che contiene il valore del canale analogico in uscita.

Nout contiene un valore da 0 a N dove N sono il numero dei canali analogici in uscita configurati nella scheda -1.

0 e' il primo canale.

Il valore `Int32` che ritorna e' compreso da 0 a valore Max del canale analogico (4096).

Attualmente i canali analogici in uscita sono disponibili solo per scheda NG35.

4.4.3 `Int32 CnIO.ReadChEncoder(Int32 Canale)`

Ritorna un `Int32` che contiene il valore del canale encoder letto.

Canale contiene un valore da 0 a N dove N sono il numero dei canali encoder configurati nella scheda -1. 0 e' il primo canale.

Il valore `Int32` che ritorna e' compreso in un `Int32` e rappresenta il conteggio degli impulsi encoder attuali. Attualmente i canali encoder sono disponibili solo per scheda NG35.

4.4.4 `Int32 CnIO.ReadDigitalInput(Int32 Ninput)`

Ritorna un `Int32` che contiene lo stato dell' ingresso digitale letto

0 --> ingresso disattivo

1 --> ingresso attivo

Ninput e' un valore da 0 a N dove N sono il numero massimo di ingressi digitali presenti sulla scheda -1

4.4.5 `Int32 CnIO.ReadDigitalOut(Int32 Nout)`

Ritorna un `Int32` che contiene lo stato dell' uscita digitale letta

0 --> uscita disattiva

1 --> uscita attiva

Nout contiene un valore da 0 a N dove N sono il numero massimo di uscite digitali presenti sulla scheda -1.

4.4.6 `Int32 CnIO.ReadGroupInputDig(Int32 Group)`

Ritorna un `Int32` che contiene lo stato del gruppo di 32 ingressi digitali letto

Gli ingressi sono mappati a Bit, pertanto il relativo Bit a 1 indica l' ingresso attivo, viceversa indica l' ingresso disattivo.

Esempio c#:

```
private void TestInput()
{
    // legge il primo gruppo di ingressi
    Int32 Group = MyIso.CnIO.ReadGroupInputDig(0);
    // controlla se ingresso 1 e 5 attivi
    if((Group & 1)==1 && (Group & 16)==16)
        // ingresso 1 e 5 attivo
}
```

4.4.7 **Int32 CnIO.ReadGroupOutDig(Int32 Group)**

Ritorna un Int32 che contiene lo stato del gruppo di 32 uscite digitali lette
Le uscite sono mappati a Bit, pertanto il relativo Bit a 1 indica l'uscita attiva, viceversa indica l'uscita disattiva.

Esempio c#:

```
private void TestOut()
{
    // legge il primo gruppo di uscita
    Int32 Group = MyIso.CnIO.ReadGroupOutDig(0);
    // controlla se uscita 2 e 6 attive
    if((Group & 2)==2 && (Group & 32)==32)
        // uscita 2 e 6 attive
}
```

4.4.8 **Int32 CnIO.ReadQuotaReal(Int32 Asse)**

Ritorna un Int32 che contiene il valore della quota asse Reale indicata (se abilitate). L'unita' di misura della quota e' quella definita nel file di configurazione dal parametro **RESQUOTE**.

RESQUOTE=1000 (millesimi di millimetro)

il valore letto e' in millesimi di millimetro

RESQUOTE=10000 (decimillesimi di millimetro)

il valore letto e' in decimillesimi di millimetro

Le quote assi REALI sono le quote comprensive dell'errore di spazio, pertanto il loro valore puo' discostarsi da quelle TEORICHE calcolate dall'interpolatore.

Asse e' un valore che va da 0 a N dove N e' il numero massimo di assi configurati sul CN-1.

4.4.9 **Int32 CnIO.ReadQuotaTeor(Int32 Asse)**

Ritorna un Int32 che contiene il valore della quota asse TEORICA indicata. L'unita' di misura della quota e' quella definita nel file di configurazione dal parametro **RESQUOTE**.

Es: **RESQUOTE=1000** (millesimi di millimetro)

il valore letto e' in millesimi di millimetro

RESQUOTE=10000 (decimillesimi di millimetro)

il valore letto e' in decimillesimi di millimetro

Le quote assi TEORICHE le quote calcolate dall'interpolatore e sono escluse dall'eventuale errore di spazio commesso dall'asse.

Asse e' un valore che va da 0 a N dove N e' il numero massimo di assi configurati sul CN-1.

4.4.10 **Void CnIO.SetResDigitalOut(Int32 Nout,Int32 Stato)**

Setta/Resetta l'uscita digitale indicata in **Nout**.

Stato=0 Uscita resettata

Stato=1 uscita settata

Nout e' un valore che va da 0 a N dove N e' il numero di uscite digitali del CN -1.

4.4.11 **Void CnIO.WriteAnalogOut(Int32 Nout,Int32 Valore)**

Scriva l'uscita analogica sul CN.

Nout e' un valore che va da 0 a N dove N e' il numero massimo di uscite configurate -1.

Valore Contiene un Int32 che va da 0 a 4096

Attualmente la funzione e' disponibile solo per NG35.

4.4.12 **Byte[]** CnIO.ReadByteCncMemory(**Int32** Addr,**Int32** Len)

Legge la memoria del CNC in Byte

Addr Indirizzo di lettura memoria

Len Lunghezza in bytes da leggere

Ritorna un array di bytes pari alla lunghezza Len

4.4.13 **Int32[]** CnIO.ReadInt32CncMemory(**Int32** Addr,**Int32** Len)

Legge la memoria del CNC in Int32

Addr Indirizzo di lettura memoria

Len Lunghezza in bytes da leggere

Ritorna un array di Int32 pari alla lunghezza Len

4.4.14 **Void** CnIO.WriteByteCncMemory(**Int32** Addr,**Byte[]** Data)

Scrive la memoria del CNC in Byte

Addr Indirizzo di scrittura memoria

Data Array di bytes da scrivere

4.4.15 **Void** CnIO.WriteInt32CncMemory(**Int32** Addr,**Int32[]** Data)

Scrive la memoria del CNC in Int32

Addr Indirizzo di scrittura memoria

Data Array di Int32 da scrivere

4.5 Config

Racchiude tutti i metodi e proprietà che gestiscono la configurazione del CN.

La configurazione viene letta dal file `IsoNs.cfg`.

Per utilizzare questa classe occorre aggiungere i riferimenti alla **ComSynk.dll**.

Questa risulta essere una classe che viene utilizzata solo per scopi precisi. Peraltro la descrizione dei vari metodi e proprietà prevede la conoscenza del CN.

4.5.1 **Config.GetCncVersion** proprietà di tipo `String` – ReadOnly

Ritorna Una stringa che contiene la versione dell' applicazione residente sul CNC.

4.5.2 **Config.NsConfig.AddressFixed** proprietà di tipo `Int32` - ReadOnly

Ritorna l' indirizzo delle Variabili Fixed della CPU VIRTUALE SU PC.

Le variabili Fixed sono un gruppo di variabili che servono per lo scambio dati con il CN peraltro la modifica del loro valore può compromettere il funzionamento.

4.5.3 **Config.NsConfig.DebugMode** proprietà di tipo `Boolean` - ReadOnly

Ritorna True se IsoNs in modo debug

Configurazione in modo Debug. Non trasferisce i parametri al CN utilizzando quelli di default.

Per abilitare **DebugMode** occorre inserire il parametro **INSERTNLINIA** del file `IsoNs.cfg` ad un valore di **2**. Il modo DEBUG è utile solo in fase di sviluppo applicazioni.

4.5.4 **Config.NsConfig.FullDebug** proprietà di tipo `Boolean` - ReadOnly

Ritorna True se IsoNs ha abilitato il **FULLDEBUG**, cioè la possibilità di gestire i BreakPoint.

Per abilitare il FullDebug occorre inserire il parametro **DEBUGMODE** del file `IsoNs.cfg` a **SI**.

Viceversa **NO** per disabilitare.

4.5.5 **Config.NsConfig.GetBaudRateRs232** proprietà di tipo `Int32` - ReadOnly

Ritorna il baud rate della porta di comunicazione RS232 selezionata.

Il baudrate fa riferimento al parametro **COMRS232** del file `IsoNs.cfg`.

ATTENZIONE!!!

Nessun controllo viene utilizzato per gestire valori non standard di comunicazione con la porta RS232.

Il valore di comunicazione deve corrispondere con quello impostato sul CN.

4.5.6 `String[]` **Config.NsConfig.GetCnError(Int32[] Alarm)**

Ritorna Gli allarmi sul CN.

Tale metodo viene utilizzato da `CnError.GetError()`.

4.5.7 `String` **Config.NsConfig.GetCodePause(Int32 Ncode)**

Ritorna la descrizione di pausa configurata nel file `IsoNs.cfg` nella sezione **#CODEPAUSE**

Il valore **Ncode** viene passato dal relativo evento `PauseChanged`.

Esempio c#:

```
private void PauseChanged(object sender, NsWork.PauseArgs e)
{
    if (e.Value == false)
    {
        // CN in ripresa da pausa
    }
    else
    {
        // CN in in pausa in e.CodePause il codice di pausa
        DescrPausa.Text =IsoNs.Config.NsConfig.GetCodePause (e.CodePause);
    }
}
```

4.5.8 **Config.NsConfig.GetComType** proprietà di tipo **String** - ReadOnly

Ritorna il tipo di comunicazione attivato con il CN

"ETH" Comunicazione tramite porta Ethernet, pertanto occorre fare riferimento ai parametri **IpAddr** (indirizzo IP del CN NG) e **Port** porta di comunicazione (default 6000).

"RS232" Comunicazione tramite porta RS232, pertanto occorre fare riferimento ai parametri **GetBaudRateRs232** (baud rate della porta) e **GetNumComRs232** porta di comunicazione COM selezionata su PC

Il tipo di comunicazione viene gestito dal parametro **COMTYPE** del file **IsoNs.cfg**

4.5.9 **Config.NsConfig.GetCpuType** proprietà di tipo **String** - ReadOnly

Ritorna il tipo di CPU utilizzata per IsoNs

"VIRTUAL" CPU virtuale su PC (il part program viene eseguito da una CPU virtuale ISONS che utilizza le risorse presenti sul PC)

"REAL" CPU reale su NG (il partprogram viene eseguito dalla CPU della scheda NG)
NON ANCORA ABILITATO

Il tipo CPU viene gestito dal parametro **CPUTYPE** del file **IsoNs.cfg**

4.5.10 **Config.NsConfig.GetDefine** proprietà di tipo **List<String[]>** - ReadOnly

Ritorna la lista delle DEFINE configurate. Questo parametro e' gestito dal COMPILATORE in modo automatico

4.5.11 **String Config.NsConfig.GetDefineError(Int32 Nerror)**

Ritorna la lista di errori configurati.

Tale metodo viene utilizzato da **CnError.GetError()**.

4.5.12 **Config.NsConfig.GetDimPianoX** proprietà di tipo **Int32** - ReadOnly

Ritorna le dimensioni X in mm del piano di lavoro impostate

Questo parametro puo' risultare utile per eventuali controlli.

Attualmente viene gestito da **Simulation.dll**

Le dimensioni fanno riferimento al parametro **DIMPIANO** del file **IsoNs.cfg**

4.5.13 **Config.NsConfig.GetDimPianoY** proprietà di tipo **Int32** - ReadOnly

Vedi **GetDimPianoX**

4.5.14 **Config.NsConfig.GetDimPianoZ** proprietà di tipo **Int32** - ReadOnly

Vedi **GetDimPianoX**

4.5.15 **Config.NsConfig.GetGottimizzate** proprietà di tipo **Boolean** - ReadOnly

Parametro esclusivo per il compilatore.

Se True vengono gestite le funzioni Gin modo ottimizzato.

L' impostazione a True deve essere effettuata esclusivamente se file ISO che non utilizzano gestione estesa tipo cicli IF Loop ecc.

GetGottimizzate fa riferimento al parametro **GOTTIMIZZATE** del file **IsoNs.cfg**

4.5.16 **Config.NsConfig.GetInsertNlinea** proprietà di tipo **Boolean** - ReadOnly

Parametro esclusivo per il compilatore.

Se **True** vengono inseriti i numeri di linea nel codice PCODE

Cambiare questo parametro puo' precludere alcuni funzionamenti.

Il valore a False viene inserito solamente per lavorazioni speciali da verificare sulla macchina

GetInsrtNlinea fa riferimento al parametro **INSERTNLINIA** del file **IsoNs.cfg**

4.5.17 **String Config.NsConfig.GetInternalError(Int32 Nerror)**

Ritorna la descrizione dell' errore interno
Tale metodo viene utilizzato da **CnError.GetError()**.

4.5.18 **Config.NsConfig.GetMacroError** proprietà di tipo **Int32- ReadOnly**

Parametro esclusivo per il compilatore.

Ritorna il numero di M da attivare al verificarsi di un errore sul CN o sul PartProgram
Gestito dalla sezione **#DEF MACRO->ERROR** del file **IsoNs.cfg**

4.5.19 **Config.NsConfig.GetMacroGoBlock** proprietà di tipo **Int32- ReadOnly**

Parametro esclusivo per il compilatore.

Ritorna il numero di M da attivare per l' esecuzione del PartProgram da ricerca Blocco
Gestito dalla sezione **#DEF MACRO->GOBLOCK** del file **IsoNs.cfg**

4.5.20 **Config.NsConfig.GetMacroGoPause** proprietà di tipo **Int32- ReadOnly**

Parametro esclusivo per il compilatore.

Ritorna il numero di M da attivare quando il CN passa da uno stato di **PAUSA** a uno di **RUN**
Gestito dalla sezione **#DEF MACRO->GOPAUSE** del file **IsoNs.cfg**

4.5.21 **Config.NsConfig.GetMacroGoRetrace** proprietà di tipo **Int32- ReadOnly**

Parametro esclusivo per il compilatore.

Ritorna il numero di M da attivare quando il CN fa una ripartenza da retrace
Gestito dalla sezione **#DEF MACRO->GORETRACE** del file **IsoNs.cfg**

4.5.22 **Config.NsConfig.GetMacroPause** proprietà di tipo **Int32- ReadOnly**

Parametro esclusivo per il compilatore.

Ritorna il numero di M da attivare quando il CN passa da uno stato di **RUN** a uno di **PAUSA**
Gestito dalla sezione **#DEF MACRO->PAUSE** del file **IsoNs.cfg**

4.5.23 **Config.NsConfig.GetMacroStop** proprietà di tipo **Int32- ReadOnly**

Parametro esclusivo per il compilatore.

Ritorna il numero di M da attivare quando il CN passa da uno stato di **RUN** a uno di **STOP**
Gestito dalla sezione **#DEF MACRO->STOP** del file **IsoNs.cfg**

4.5.24 **Config.NsConfig.GetNtab** proprietà di tipo **Int32- ReadOnly**

Parametro esclusivo per il compilatore.

Ritorna il numero il numero di tabelle utensili presenti
Gestito dalla sezione **#TABUT** del file **IsoNs.cfg**

4.5.25 **Config.NsConfig.GetNumComRs232** proprietà di tipo **Int32- ReadOnly**

Ritorna il numero il numero di tporta RS232 configurata per la comunicazione con il CN
Gestito dal parametro **COMRS232** del file **IsoNs.cfg**

4.5.26 **Config.NsConfig.GetNvarM** proprietà di tipo **Int32- ReadOnly**

Parametro esclusivo per il compilatore.

Ritorna il numero il numero di variabili per funzioni M riservate dal compilatore
Gestito dal parametro **NVAR_M** del file **IsoNs.cfg**

4.5.27 **Config.NsConfig.GetParMac** proprietà di tipo **List<ComSynk.ParametriMacchina>- ReadOnly**

Ritorna la lista dei parametri macchina del CN

Tale metodo viene utilizzato dalla classe **Param**

Gestito dalla sezione **#PARMAC** del file **IsoNs.cfg**

4.5.28 **Int32[] Config.NsConfig.GetParTab(Int32 Ntab)**

Ritorna una array di Int32 con lunghezza 20 che rappresenta i parametri della tabella utensile indirizzata in Ntab. L'istruzione **Tn** del part program attiva automaticamente questi parametri.
I parametri fanno riferimento alla seguente descrizione:

1 → **Diametro utensile**

2 → **Lunghezza Utensile**

3 → **Vel Rot utensile**

4,5,6,7,8,9,10,11,12 **offset per la testa X,Y,Z,A,B,C,U,V,W**

altri user define

Gestito dalla sezione **#TABUT** del file **IsoNs.cfg**

4.5.29 **Config.NsConfig.GetPathIsoNs**

proprietà di tipo **String**- ReadOnly

Ritorna la path dove e' installato IsoNs

4.5.30 **Config.NsConfig.GetRunErrComp**

proprietà di tipo **Boolean**- ReadOnly

Se **True** viene eseguito il PART PROGRAM anche se contiene errori di compensazione utensile
Se **False** non permette l' esecuzione del PartProgram

Gestito dal parametro **RUN_ERR_COMP** del file **IsoNs.cfg**

4.5.31 **Config.NsConfig.GetSeqHoming**

proprietà di tipo **Int32[]**- ReadOnly

Ritorna la sequenza configurata per l' homing degli assi in base a quelli che devono effettuare l' homing.
ES:

ASSI CONFIGURATI X,Y,Z

Sequenza configurata 2,1

In questo caso **GetSeqHoming** ritorna un array di Int32 di lunghezza 2 con i seguenti Valori:

Int32[0]=2

Int32[1]=1

Che stanno a significare che il primo ad effettuare l' homing e' l' asse Z, il secondo l' asse Y mentre l' ase X non fa l' homing.

Questo parametro serve esclusivamente quando si effettua un Homing degli assi automatico.

Gestito dal parametro **HOMINGASSI** del file **IsoNs.cfg**

4.5.32 **Config.NsConfig.GetTimeOutRs232**

proprietà di tipo **Int32**- ReadOnly

Parametro esclusivo per ComSynk.dll

Ritorna il TimeOut in millisecondi impostato per errori comunicazione RS232

Gestito dal parametro **TIMEOUTRS232** del file **IsoNs.cfg**

4.5.33 **Int32 Config.NsConfig.ValueParByName(String Name)**

Ritorna il valore dl parametro macchina passato in **Name**

4.5.34 **Config.NsConfig.GruppiAllarm**

proprietà di tipo **Int32**- ReadOnly

Ritorna il numero di gruppi di 32 Bit di allarmi configurati da leggere dal CN

Tale proprietà viene utilizzato da **ComSynk.dll**.

4.5.35 **Config.NsConfig.GruppiInput**

proprietà di tipo **Int32**- ReadOnly

Ritorna il numero di gruppi a 32 bit di ingressi digitali configurati da leggere dal CN

Tale proprietà viene utilizzato da **ComSynk.dll**.

4.5.36 **Config.NsConfig.IpAddr**

proprietà di tipo **String**- ReadOnly

Ritorna l' indirizzo IP configurato per connessione ETHERNET con il CN

Tale proprietà viene utilizzato da **ComSynk.dll**.

Gestito dal parametro **IPADDR** del file **IsoNs.cfg**

4.5.37 **Void Config.NsConfig.LoadConfigura()**

Carica il file **IsoNs.cfg**.

Tale metodo e' automaticamente chiamato da NsWork.dll

4.5.38 **Config.NsConfig.MaxCodeMemory** proprietà di tipo **Int32**- ReadOnly

Ritorna il numerodi Byte riservati per la memoria del PartProgram

Gestito dal parametro **CODEMEMORY** del file **IsoNs.cfg**

4.5.39 **Config.NsConfig.MaxFeed** proprietà di tipo **Int32**- ReadOnly

Ritorna valore massimo della **F** impostabile

Gestito dal parametro **FEEDMAX** sezione **#PARMAC** del file **IsoNs.cfg**

4.5.40

4.5.41 **Config.NsConfig.MaxSpeed** proprietà di tipo **Int32**- ReadOnly

Ritorna valore massimo della **S** impostabile

Gestito dal parametro **SPEEDMAX** sezione **#PARMAC** del file **IsoNs.cfg**

4.5.42 **Config.NsConfig.MaxVper** proprietà di tipo **Int32**- ReadOnly

Ritorna il numero di divisioni massime gestite dal convertitore analogico/digitale per la lettura del potenziometro di Override

Gestito dal parametro **MAXVPER** del file **IsoNs.cfg**

4.5.43 **Config.NsConfig.Nassi** proprietà di tipo **Int32**- ReadOnly

Ritorna il numero assi configurati sul CN

Gestito dal parametro **NASSI** del file **IsoNs.cfg**

4.5.44 **Config.NsConfig.NdecimalFeed** proprietà di tipo **Int32**- ReadOnly

Ritorna il numero di decimali per rappresentare su eventuale interfaccia della velocità assi **F** impostatata.

Tale parametro serve esclusivamente ad una rappresentazione a video del valore della **F**

Gestito dal parametro **NDECIMALFEED** del file **IsoNs.cfg**

4.5.45

4.5.46 **Config.NsConfig.NdecimalQuote** proprietà di tipo **Int32**- ReadOnly

Ritorna il numero di decimali per rappresentare su eventuale interfaccia il valore delle quote assi.

Tale parametro serve esclusivamente ad una rappresentazione a video **delle quote assi**

Gestito dal parametro **NDECIMALQUOTE** del file **IsoNs.cfg**

4.5.47 **Config.NsConfig.NdecimalSpeed** proprietà di tipo **Int32**- ReadOnly

Ritorna il numero di decimali per rappresentare su eventuale interfaccia della velocità mandrino **S** impostatata.

Tale parametro serve esclusivamente ad una rappresentazione a video del valore della **S**

Gestito dal parametro **NDECIMALSPEED** del file **IsoNs.cfg**

4.5.48 **Config.NsConfig.NomiAssi** proprietà di tipo **String[]**- ReadOnly

Ritorna un array String che rappresenta il nome degli ASSI configurati

Gestito dal parametro **NOMEASSI**del file **IsoNs.cfg**

4.5.49 Config.NsConfig.NunitaFeed proprietà di tipo **Int32**- ReadOnly
Ritorna il numero di unita' per rappresentare su eventuale interfaccia la velocità assi **F** impostata.
Tale parametro serve esclusivamente ad una rappresentazione a video del valore della F

Gestito dal parametro **NUNITAFEED** del file **IsoNs.cfg**

4.5.50 Config.NsConfig.NunitaQuote proprietà di tipo **Int32**- ReadOnly
Ritorna il numero di unita' per rappresentare su eventuale interfaccia il valore delle quote assi.
Tale parametro serve esclusivamente ad una rappresentazione a video delle quote assi

Gestito dal parametro **NUNITAQUOTE** del file **IsoNs.cfg**

4.5.51 Config.NsConfig.NunitaSpeed proprietà di tipo **Int32**- ReadOnly
Ritorna il numero di unita' per rappresentare su eventuale interfaccia la velocità mandrino **S** impostata.
Tale parametro serve esclusivamente ad una rappresentazione a video del valore della S

Gestito dal parametro **NUNITASPEED** del file **IsoNs.cfg**

4.5.52 Config.NsConfig.Port proprietà di tipo **Int32**- ReadOnly
Ritorna il numero di porta ETHERNET impostata per la comunicazione con il CN (default 6000)

Gestito dal parametro **PORT** del file **IsoNs.cfg**

4.5.53 Config.NsConfig.ReadInput proprietà di tipo **Int32[]** - ReadOnly
Ritorna un array di Int32 che rappresenta gli ingressi digitali abilitati alla generazione di eventi
Tale proprietà viene utilizzato da **ComSynk.dll**.

Gestito dal parametro **READ_INPUT** del file **IsoNs.cfg**

4.5.54 Config.NsConfig.SogliaInsArchi proprietà di tipo **Double**- ReadOnly
Rappresenta il valore di soglia per inserimento ARCHI durante la compensazione utensile
Tale proprietà viene utilizzato da ComSynk.dll.

Gestito dal parametro **SOGLIA_ARCHI_COMP** del file **IsoNs.cfg**

4.5.55 Config.NsConfig.TimeOut proprietà di tipo **Int32**- ReadOnly
Parametro esclusivo per **ComSynk.dll**
Ritorna il TimeOut in millisecondi impostato per errori comunicazione Ethernet

Gestito dal parametro **TIMEOUTETH** del file **IsoNs.cfg**

4.5.56 Config.NsConfig.TimeOutEnable proprietà di tipo **Int32**- ReadOnly
Parametro esclusivo per **Homing.dll**
Ritorna il TimeOut in millisecondi impostato per abilitazione ASSI

Gestito dal parametro **TIMEOUTENABLE** del file **IsoNs.cfg**

4.5.57 Config.NsConfig.TimeOutHome proprietà di tipo **Int32**- ReadOnly
Parametro esclusivo per **Homing.dll**
Ritorna il TimeOut in millisecondi impostato per Homing ASSI

Gestito dal parametro **TIMEOUTHOMED** del file **IsoNs.cfg**

4.5.58 **Config.NsConfig.TimerScanner**

proprietà di tipo **Int32**- ReadOnly

Parametro esclusivo per **ComSynk.dll**

Ritorna il tempo in millisecondi di scansione del timer generale (lettura quote, generazione eventi, ecc.)

Gestito dal parametro **TIMERSCANNER** del file **IsoNs.cfg**

4.5.59 **Config.NsConfig.UserGeneric**

proprietà di tipo **Int32**- ReadOnly

Parametro esclusivo per **ComSynk.dll**

Ritorna il numero di variabili UserGeneric da scambirasi con il CN

Gestito dal parametro **READ_USER** del file **IsoNs.cfg**

4.5.60 **Config.NsConfig.VeryAddressFixed**

proprietà di tipo **Int32** - ReadOnly

Ritorna l' indirizzo fisico delle Variabili Fixed del CN.

Le variabili Fixed sono un gruppo di variabili sche servono per lo scambio dati con il CN perntato la modfica del loro valore puo' compromettere il funzionamento.

L' indirizzo delle Fixed viene gestito dal parametro di configurazione **ADDRESSFIXED** del file **IsoNs.cfg**.

Tale indirizzo dipende dall' hardware utilizzato (NG35 o NGM13).

L' indirizzo ASSOLUTO sulla scheda Ngxx e' composto dal valore base letto con **AddressFixed** sommandoci il **NUMEROFIXED*4** relativo alla fixed che vogliamo legger/scrivere sul CN.

Es:

Acesso alla FIXED 0

ADDRESSFIXED0=Config.NsConfig.AddressFixed +(0*4)

Acesso alla FIXED 1

ADDRESSFIXED1=Config.NsConfig.AddressFixed +(1*4)

Config.NsConfig.VisuaZeri(proprietà di tipo **Boolean- ReadOnly)**

Ritorna true se abilitata la visualizzazione degli zeri prima delle quote assi.

Tale parametro serve esclusivamente ad una rappresentazione a video delle quote assi

Gestito dal parametro **VISUAZERI** del file **IsoNs.cfg**

4.6 ExecuteMtoCn

Racchiude tutti i metodi e proprietà che gestiscono l' esecuzione delle funzioni M sul CN
Prima di eseguire una funzione M sul CN questa deve essere stata scritta in CODICE VTB nell' applicazione installata.

4.6.1 Boolean GoMtoCn(Int32 CodeM, Int32[] Param)

Metodo che esegue una M sul CN.

CodeM Codice della M da eseguire

Param Array di Int32 di parametri da scambiare per la M (il numero massimo di parametri scambiati con le M e' definito in configurazione da NVARM)

Ritorna **True** M eseguita

False M non configurata sul CN

Per attendere l' effettiva terminazione della M occorre gestire l' evento **MexecuteChanged**

Esempio c#:

```
private void EseguiM()
{
    Int32[] Param=new Int32[3]; // scambia tre parametri con il CN
    // assegna i valori ai parametri
    Param[0]=10;
    Param[1]=20;
    Param[2]=30;
    // attiva la M100
    if(IsoNs.ExecuteMtoCn.GoMtoCn(100, Param)==false)
        // errore esecuzione M
}
```

4.6.2 Int32 ReadParM(Int32 Npar)

Metodo che legge il parametro M indicato dal CN scritto da **WriteParM** o **GoMtoCn**

Ritorna il valore del parametro letto

Genera un' eccezione se parametro inesistente

4.6.3 Boolean WriteParM(Int32 Npar,Int32 Valore)

Metodo che scrive un parametro M sul CN

Ritorna True parametro scritto regolarmente

Tale metodo puo' anche essere utilizzato per scrivere le variabili definite dal compilatore \$_PARAM1...

```
// scrive il parametro 1 2 della M
IsoNs.ExecuteMtoCn.WriteParM(0,100);
IsoNs.ExecuteMtoCn.WriteParM(1,200);
// legge i parametri da part program
// spostando gli assi a param1 e 2
G1X[$_PARAM1]Y[$_PARAM2]
```

4.6.4 Void StopCnMacro()

Metodo che sospende tutte le M attive sul CN.

L' effettiva sospensione deve essere effettuata dall' applicazione VTB testando la variabile:

ISOV1_STATUS_M_STOP

Quando questa variabile si porta al valore 1 è necessario tramite la stessa applicazione VTB terminare le M in corso. La variabile **ISOV1_STATUS_M_STOP** può essere resettata dalla stessa applicazione una volta acquisita:

Es: Vtb

```
if ISOV1_STATUS_M_STOP=1
    'stop da iso
    'Stop anubatrice
    StopAllMacro()
    ISOV1_STATUS_M_RUN=0    ' sblocca il CN
    ISOV1_STATUS_M_STOP=0
endif
```

4.7 GestFunzMHM

Racchiude tutti i metodi e proprietà che gestiscono la compilazione di M e HM che vengono eseguite su PC (non all' interno del CN)

Queste possono essere poi richiamate dal metodo `ExecuteMtoCn.GoMtoCn(...)`

4.7.1 Boolean GenerateHM (Int32 Num, String Codelso, out NsWork.IsoNs.ErrorCompiler[] Err)

4.7.2 Boolean GenerateM (Int32 Num, String Codelso, out NsWork.IsoNs.ErrorCompiler[] Err)

Questo metodo permette di compilare un funzione HM o M e salvarla automaticamente nella cartella di configurazione.

Num Numero della funzione M o HM con il quale deve essere richiamata

Codelso Stringa del codice ISO

Err Array di `NsWork.IsoNs.ErrorCompiler` dove ritornano gli errori di compilazione

Ritorna **True** M/HM generata regolarmente

False errore

Esempio c#:

```
// esempio di generazione funzione Mo Hm
private void GeneraMHM()
{
    NsWork.IsoNs.ErrorCompiler[] Err;
    if(IsoNs.GestFunzMHM.GenerateM(10, "G1X100Y100Z0\nG4F2\nG1X0Y0Z0", out Err)==false)
    {
        // check errori di compilazione
        for (int n = 0; n < Err.Length; n++)
            Label1.Text += "E:" + Err[n].Nlinea + " " + Err[n].TipoErr;
    }
    // la M o la HM e' disponibile con il codice 10
}
```

4.8 Homing

Racchiude tutti i metodi e proprietà che gestiscono l' homing e l' abilitazione degli assi

4.8.1 Void EnableAxis(Int32 Naxis, Boolean Stato)

Questo metodo permette di abilitare o disabilitare un asse.
Quando l' asse e' abilitato, il relativo driver entra in controllo del motore

Naxis Numero della asse da abilitare/disabilitare

Stato **True** Abilita – **False** Disabilita

Il numero dell' asse fa riferimento all' indice degli assi configurati
In questa fase e' possibile gestire il parametro di TIMEOUTENABLE

Esempio c#:

```
// evento click singola abilitazione/disabilitazione nel TAG del button indice dell' asse
void Enable_Click(object sender, EventArgs e)
{
    Control Tp;
    Tp = (Button)sender;
    Index=(Int32)Tp.Tag;
    if (IsoNs.StatusCn.IsStatusEnableAxis[Index] == true)           // disabilita
        IsoNs.Homing.EnableAxis(Index, false);
    else
        IsoNs.Homing.EnableAxis(Index, true); // abilita asse
    // imposta il time out
    timerTimeOutEnable.Interval = IsoNs.Config.NsConfig.TimeOutEnable;
    timerTimeOutEnable.Enabled = true; // abilita il timer per gestione errore
}

// time out su abilitazione asse
private void timerTimeOutEnable_Tick(object sender, EventArgs e)
{
    timerTimeOutEnable.Enabled = false;
    System.Windows.Forms.MessageBox.Show("ASSE " + IsoNs.Homing.GetNomiAssi[Index] + " NON
    RISPONDE AL COMANDO", "ENABLE/DISABLE ERROR !!!",
    System.Windows.Forms.MessageBoxButtons.OK, System.Windows.Forms.MessageBoxIcon.Error);
}
```

4.8.2 GetNomiAssi proprietà di tipo **String[]**- ReadOnly

Ritorna un array String che rappresenta il nome degli ASSI configurati

4.8.3 GetSeqHoming proprietà di tipo **Int32[]**- ReadOnly

Ritorna la sequenza configurata per l' homing degli assi in base a quelli che devono effettuare l' homing.
ES:

ASSI CONFIGURATI X,Y,Z

Sequenza configurata 2,1

In questo caso **GetSeqHoming** ritorna un array di Int32 di lunghezza 2 con i seguenti Valori:

Int32[0]=2

Int32[1]=1

Che stanno a significare che il primo ad effettuare l' homing e' l' asse Z, il secondo l' asse Y mentre l' ase X non fa l' homing.

4.8.4 Void StartHoming(Int32 Naxis)

Questo metodo abilita la fase di ricerca di homing dell' asse indirizzato
La procedura di ricerca di homing dipende da come viene gestita dall' applicazione VTB

Naxis Numero della asse

Il numero dell' asse fa riferimento all' indice degli assi configurati
In questa fase e' possibile gestire il parametro di TIMEOUTHOME

Esempio c#:

```
// evento click su home nel TAG del button indice dell' asse
void Home_Click(object sender, EventArgs e)
{
    Control Tp;
    Tp = (Button)sender;
    Index=(Int32)Tp.Tag;
    if (IsoNs.StatusCn.IsStatusEnableAxis[Index] == true)
        IsoNs.Homing.StartHoming(Index);
    // imposta il time out
    timerTimeOutHome.Interval = IsoNs.Config.NsConfig.TimeOutHome
    timerTimeOutHome.Enabled = true; // abilita il timer per gestione errore
}

// time out su homing asse
private void timerTimeOutHome_Tick(object sender, EventArgs e)
{
    timerTimeOutHome.Enabled = false;
    System.Windows.Forms.MessageBox.Show("ASSE " +IsoNs.Homing.GetNomiAssi[Index] + " NON
    RISPONDE AL COMANDO", "HOMING ERROR !!!",
    System.Windows.Forms.MessageBoxButtons.OK, System.Windows.Forms.MessageBoxIcon.Error);
}
```

4.8.5 Void StopHoming()

Interrompe la sequenza di homing in corso arrestando l' asse nel punto attuale con un comando STOP.

4.8.6 Int32 ReadSfasaTacca(Int32 Naxis)

Metodo che legge lo sfasamento in impulsi encoder della tacca di zero rispetto al micro di homing.
Il valore viene aggiornato dopo la ricerca di zero asse.
Tale metodo è utilizzabile solo per assi che hanno encoder con tacca di zero collegata sul CN
Genera un'eccezione se asse inesistente

4.8.7 Void PresetAbsEnc(Int32 Naxis)

Metodo che esegue il preset di ZERO per encoder di tipo assoluto multigioco.
Tale metodo fissa sul CN la posizione attuale come ZERO. Il metodo deve essere utilizzato quando si vuole fissare lo ZERO ASSOLUTO per la prima volta.

4.9 JogAxis

Racchiude tutti i metodi e proprietà che gestiscono la movimentazione manuale degli assi

4.9.1 AbsRel proprietà di tipo **Boolean**- Read/Write

Legge o imposta la movimentazione assoluta o relativa

False movimentazione assoluta

True movimentazione relativa

Questa determina come deve agire il metodo **MoveAxis**.

Nel caso di movimentazione assoluta impostata, il valore del Target è riferito allo ZERO MACCHINA.

Nel caso di movimentazione relativa impostata, il valore del Target è riferito al punto dove si trova l'asse

4.9.2 ExtOverride proprietà di tipo **Boolean**- Read/Write

Abilita o disabilita il controllo del potenziometro esterno di override

False disabilita potenziometro esterno

True abilita potenziometro esterno

Con potenziometro esterno disabilitato. La percentuale di FEED impostata è pari al 100%

4.9.3 FeedAxis proprietà di tipo **Int32**- Read/Write

Imposta/Legge la velocità percentuale di movimentazione assi in manuale

Il valore deve essere compreso tra 0 e 100

Genera eccezione per valore fuori Range

4.9.4 Void Jog(Boolean Direction)

Metodo che attiva il JOG dell'asse selezionato con **SelectAxisForJog** nella direzione passata.

Direction – True/False Direzione di movimento dell'asse ORARIO/ANTIORARIO

Comunque la direzione di movimentazione dipende dalla tipologia dell'asse e dalla sua configurazione all'interno del Driver

ATTENZIONE!!!

Per interrompere il movimento utilizzare il metodo **StopMove()**

Esempio c#:

```
// evento button premuto
private void button1_MouseDown(object sender, MouseEventArgs e)
{
    IsoNs.JogAxis.Jog(true);
}

// evento button rilasciato
private void button1_MouseUp(object sender, MouseEventArgs e)
{
    IsoNs.JogAxis.StopMove();
}
```

4.9.5 Void MoveAxis(Double Target, Int32 Axis)

Metodo che muove l' asse indicato in **Axis** alla posizione **Target** passata come Double.

L' unita' di misura della posizione Target e' quella impostata nella condifurazione.

Se il CN si trova in movimentazione **RELATIVA AbsRel=True** la posizione Target viene considerata come spazio percorso dal punto dove si trova l' asse.

Se il CN si trova in movimentazione **ASSOLUTA AbsRel=False** la posizione Target viene considerata dallo **ZERO MACCHINA** impostato per l' asse in movimentazione

Target Posizione di arrivo dell' asse (es: simpostazioni al millesimo di millimetro **RESQUOTE=1000** un **Target=2.134** assume un valore di **2 millimetri e 134 millesimi**);

Axis Indice dell' asse da movimentare

Se **Axis** ha un valore al di fuori di quelli configuarti viene generata un eccezione.

ATTENZIONE!!!

Per interrompere il movimento utilizzare il metodo **StopMove()**

4.9.6 Void SetShiftAxis(Int32 Axis, Double ValShift)

Metodo che muove l' asse indicato in **Axis** in modo Shift, cioè tipo manovella elettronica.

Il valore immesso nel parametro **ValShift**, viene sommato alla quota attuale dell' asse (anche se l' asse è in movimento). La quota asse viene data con incremento dipenedente dal parametro **TSHF_...** Questo valore indica il numero di impulsi di incremento quota per campionamento TAU impostato.

Es:

TAU=2 Ms

ValShift=100

TSHF_ =10

La quota viene raggiunta in 20 Ms

4.9.7 SelectAxisForJog

proprietà di tipo **Int32**- Read/Write

Imposta/Legge l' asse da movimentare con il metodo **Jog()**

Il valore deve essere compreso tra 0 e NASSI-1 impostati in configurazione

Se il **valore** e' al di fuori di quelli configuarti viene generata un eccezione.

4.9.8 Void SelectAxisAndMolt(Int32 QualeAsse, Int32 MoltVol)

Metodo che seleziona l' asse in JOG con moltiplicatore volante elettronico.

Il moltiplicatore deve avere i seguenti valori

1 **x1**

10 **x1**

100 **x100**

1000 **x1000**

Il metodo **SelectAxisAndMolt** è analogo all proprietà **SelectAxisForJog**, ma in più utilizza il parametro di moltiplicazione volante.

ATTENZIONE!!!

Se si utilizza un volante elettronico con moltiplicatore software è necessario utilizzare sempre questo metodo al posto della proprietà **SelectAxisForJog**.

4.9.9 Int32 ReadMolt()

Ritorna il valore impostato del moltiplicatore volante elettronico.

4.9.10 Void SopMove()

Metodo che muove interrompe la movimentazione dell' asse in manuale sia che questa venga effettuata con il metodo **Jog()** sia con il metodo **MoveAxis(...)**.

La fermata dell' asse viene effettuata con la rampa di accelerazione impostata nei parametri macchina del CN.

ATTENZIONE!!!

StopMove() non arresta gli assi in MODO EMERGENZA, e pertanto possono verificarsi delle situazioni nelle quali l' asse e' fuori controllo e pertanto e' **INDISPENSABILE OPERARE CON I METODO DI EMERGENZA PREVISTI NELLA MACCHINA** (Fungo ecc.)

4.10 MyMaster

Questa classe racchiude molti metodi che sono già descritti.

Questa permette l'accesso a delle parti che prevedono una programmazione avanzata, pertanto non è consigliabile l'utilizzo.

4.11 NewComp

Stesso discorso della classe MyMaster.

NewComp gestisce la compilazione del file ISO.

4.12 Param

Questa classe gestisce tutti i parametri macchina del CN.

Permette il trasferimento, la modifica e la manutenzione.

Tutti i parametri macchina sono salvati nel file IsoNs.cfg e devono essere trasferiti al CN seguendo la procedura descritta:

- 1) Chiamare il metodo **DownloadPara()** per aggiornare la lista interna
- 2) Chiamare il metodo **SendAllPara()**
- 3) Chiamare il metodo **UpdatePara()**

I parametri vengono rappresentati nel seguente formato:

| | |
|--------------------|--|
| Nome | Nome del parametro |
| Descrizione | Descrizione del parametro |
| Gruppo | Gruppo di appartenenza del parametro (serve per dare un raggruppamento ai parametri) |
| Valore | Valore Int32 del parametro |
| IndexCn | Indirizzo in memoria del CN (i parametri con IndexCn=-1 sono utilizzati dalla CPU virtuale del PC e pertanto non vengono trasferiti al CN) |

4.12.1 Void DownloadPara()

Legge tutti i parametri dalla configurazione IsoNs.cfg e li salva in una lista interna predisponendo gli altri metodi alla lettura di questi.

Questa deve essere in ogni caso il primo metodo da richiamare prima della gestione dei parametri.

4.12.2 Boolean GetPar(Int32 Index, out String Nome, out String Descr, out String Gruppo, out Int32 Valore, out Int32 IndexCn)

Legge un parametro macchina

Index – Indice del parametro da leggere (l'indice del parametro e' la posizione all'interno della lista, in pratica segue la posizione di come sono stati scritti nel file IsoNs.cfg)

| | |
|----------------|--|
| Nome | String ritorna il nome del parametro |
| Descr | String ritorna la descrizione del parametro |
| Gruppo | String ritorna il gruppo di appartenenza del parametro |
| Valore | Int32 ritorna il valore del parametro |
| IndexCn | Int32 ritorna l'Indirizzo sul CN del parametro |

Ritorna **True** se lettura parametro Ok

False parametro non trovato o lista non inizializzata con **DownloadPara()**

Overload

4.12.3 Boolean GetPar(String Nome, out Int32 Index,out String Descr,out String Gruppo,out Int32 Valore,out Int32 IndexCn)

Legge un parametro macchina

Nome – Nome del parametro da leggere (Il nome del parametro è quello riportato nel file IsoNs.cfg CASE SENSITIVE)

| | |
|----------------|--|
| Index | Int32 Ritorna l' indice del parametro |
| Descr | String ritorna la descrizione del parametro |
| Gruppo | String ritorna il gruppo di appartenenza del parametro |
| Valore | Int32 ritorna il valore del parametro |
| IndexCn | Int32 ritorna l' Indirizzo sul CN del parametro |

Ritorna **True** se lettura parametro Ok

False parametro non trovato o lista non inizializzata con **DownLoadPara()**

4.12.4 Gruppi proprietà di tipo `String[]`- Read Only

Ritorna un array di string che contiene il nome dei gruppi dei parametri che sono stati configurati nel file **IsoNs.cfg**

4.12.5 Npar proprietà di tipo `Int32`- Read Only

Ritorna un il numero di parametri che sono stati caricati nella lista dal metodo **DownLoadPara()**

4.12.6 Boolean RipristinaBak()

Ripristina la precedente versione del file IsoNs.cfg in seguito ad un salvataggio cin il metodo **SaveCfg()**

Ritorna **True** versione Ripristinata

False nessuna versione precedente da ripristinare

4.12.7 Void SaveCfg()

Salva nel file **IsoNs.cfg** la lista dei parametri in memoria

La versione percedente puo' essere ricaricata in seguito all' invocazione del metodo **RipristinaBak()**

4.12.8 Void SendAllPara()

Trasferisce al CN tutti i parametri della lista in memoria.

Alcuni parametri avranno effetto solo dopo l'm invocazione del metodo **UpdatePara()**

4.12.9 Void UpdatePara()

Aggiorna i parametri sul CN

Tale metodo viene chiamato in seguito a **SendAllPara()**

4.12.10 Boolean WritePara(Int32 Index, Int32 Value, Boolean WriteCn)

Overload

4.12.11 Boolean WritePara(String Name, Int32 Value, Boolean WriteCn)

Scrive un parametro macchina

| | |
|----------------|---|
| Index | Indice del parametro scrivere (l' indice del parametro e' la posizione all' interno della lista, in pratica segue la posizione di come sono stati scritti nel file IsoNs.cfg) |
| Name | Nome del parametro da scrivere (il nome deve corrispondere a quello inserito nel file IsoNs.cfg) |
| Value | Valore del parametro Int32 |
| WriteCn | Se True il parametro viene trasferito al CN (non viene comunque invocato il metodo UpdatePara()) |

Ritorna **True** Scrittura Ok

False Parametro non scritto - Index /nome errato o lista vuota

4.12.12 SetVisuaQuoteReal proprietà di tipo [NsWork.ParamCs.VisType](#)- Read/Write

Setta il tipo di visualizzazione delle quote Reali degli assi se abilitate (parametro macchina **VISUAREAL** diverso da -1)

Accetta valori di tipo :

| | |
|--|-------------------------------|
| NsWork.ParamCs.VisType.DISABLE | Quote disabilitate |
| NsWork.ParamCs.VisType.REALQUOTE | Lettura quote reali assi |
| NsWork.ParamCs.VisType.ERRORSPACE | Lettura errore di spazio assi |

4.14 ProgramRun

Questa classe gestisce l' esecuzione del programma ISO. Pertanto e' tramite i metodi di questa che viene gestito un PartProgram.

4.14.1 Boolean ExecuteScript(String Code, out NsWork.IsoNs.ErrorCompiler[] Err)

Esegue il codice IsoNs passato in Code.

Uno script si comporta diversamente da un normale PartProgram, poiche' questo non gestisce una movimentazione completa degli assi ma solo G0 e G1 e puo' essere anche eseguito quando la CPU e' in stato di PAUSA

Code Stringa che contiene il codice ISONS da eseguire

Err Tipo **NsWork.IsoNs.ErrorCompiler[]** che ritorna gli errori di compilazione del codice

Esempio C#

```
// esempio di esecuzione di un codice Script IsoNs
private void EsguiScript()
{
    NsWork.IsoNs.ErrorCompiler[] Err;
    if(IsoNs.ProgramRun.ExecuteScript("G1X100Y100Z0\nG4F2\nG1X0Y0Z0", out Err)==false)
    {
        if(Err!=null)
        {
            // check errori di compilazione
            for (int n = 0; n < Err.Length; n++)
                Label1.Text += "E:" + Err[n].Nlinea + " " + Err[n].TipoErr;
        }
        else
            // impossibile eseguire lo script probabile CPU non in PAUSA on in STOP
    }
    // il codice Script IsoNs viene eseguito
}
```

4.14.2 Void ExecuteProg(Int32 Nlinea)

Esegue il part program IsoNs precedentemente compilato e caricato.
Quindi prima di chiamare il metodo ExecuteProg occorre:

- 1) Aver compilato il file IsoNs con **CompileFileIso(String Text)**
- 2) Aver caricato il file con **LoadCode()**

Al contrario di uno Script l' esecuzione del Partprogram comprende tutte le funzionalita di IsoNs, ma non puo' essere eseguito quando la CPU e' in PAUSA.

Nlinea Numero di linea di partenza esecuzione del PartProgram
Se **Nlinea e' MAGGIORE o UGUALE a 0**, Viene considerata una PARTENZA DA BLOCCO, quindi viene eseguita l' eventuale **MGOBLOCK** configurata

4.14.3 Void ExecuteProgFromMarker(Int32[] AddrMarker,Double[] ValMarker)

Esegue il part program IsoNs attivando la reale esecuzione al raggiungimento dei MARKER indicati.

MARKER

I marker sono delle normali variabili inserite nel PartProgram.

IsoNs permette di riprendere il PartPrgram quando queste variabili hanno raggiunto un certo valore. Nella programmazione EVOLUTA, con l' utilizzo di CICLI LOOP e delle VARIABILI, la ripresa dal numero di LINEA non è sufficiente, in quanto le quote assi possono essere passate da valore delle variabili che sono detreminate da un ciclo LOOP. Utilizzando i MARKER, è possibile riprendere il PartProgram dal valore di uno di questi e non dal numero di linea, quindi è possibile discriminare la ripresa all' interno di cicli LOOP.

I MARKER vengono definiti dall' istruzione ISO **MARKER \$NOMEVAR DESCRIZIONE**

Questo permette ad IsoNs di creare una Lista dei Marker facilmente recuperabile

AddrMarker Array di Int32 che contiene l' indirizzo fisico in memoria delle variabili MARKER recuperabile con la Proprietà GetMarker, la quale ritorna un Lista di tipo Compiler.[MarkerCs](#)

ValMarker Valore che deve avere ogni singolo Marker affinché la condizione di ripartenza sia soddisfatta

Es:

Nell' esempio seguente viene definita una variabile Marker di nome **\$INC**.

Questa rappresenta nel LOOP il pezzo in esecuzione

```
MARKER $INC NUMERO PEZZI
$VAR=0
$INC=0
F5
G1X0Y0
LOOP 10
    $INC=$INC+1
    G1X200
    $VAR=$VAR+50
    GOX0Y[$VAR]
END_LOOP
```

E' possibile quindi attivare la ripresa del PartProgram quando la variabile \$INC (il MARKER) assume un certo valore.

```
private void GoMarker()
{
    // legge i MARKER presenti
    List<Compiler.MarkerCs> MyMarker = MyGoBlock.MyInterface.MyIso.GetMarker;
    Int32[] AddrMarker = new Int32[MyMarker.Count];
    // ricopia indirizzi
    for (int n = 0; n < MyMarker.Count; n++)
        AddrMarker[n] = MyMarker[n].AddrVar;
    // inserisce dei valori supponendo di avere solo MARKER $INC
    Int32[] ValMarker = new Int32[1];
    ValMarker[0] = 5; // riprende dal pezzo N.5
    // esegue il Part Program
    IsoNs.ProgramRun.ExucuteProgFromMarker(AddrMarker, ValMarker);
}
```

4.14.4 Void PauseProg()

Forza una pausa del PartProgram in esecuzione in corso
Se configurata la M di PAUSE, viene eseguita.

Quando la CPU si trova in stato di PAUSA, e' possibile eseguire Script ISO e movimentazione manuale in JOG o MeveAxis().

Da uno stato di PAUSA e' possibile ripartire dal punto esatto dove e' stata interrotta la lavorazione invocando il metodo **ExecuteProg(-1)**.

Se configurata la M di GOPAUSE, viene invocata

Per far si di ripartire in modo corretto dal puntom preciso di interruzione con la pausa e necessario configurare la M di PAUSA nel seguente modo:

```
GLOBAL $SAVEX
GLOBAL $SAVEY
G61
$SAVEX=$[Q0]// salva la quota asse X attuale
$SAVEY=$[Q1]//salva la quoaat asse Y attuale
.....
.....
.....
```

e I M di ripresa da PUASA nel seguente modo:

```
GLOBAL $SAVEX
GLOBAL $SAVEY
G0X[$SAVEX] Y[$SAVEY] // riporta gli assi alle quote salvate
G62 // attende assi fermi
.....
.....
.....
.....
```

Tali operazioni non sono indispensabili, in quanto la CPU riporta comunque gli assi sul punto interrotto, pero' in questo modo abbiamo sotto controllo la posizione esatta.

4.14.5 StepMode proprietà di tipo Boolean- Read/Write

Imposta o legge l' esecuzione STEP MODE del PART PROGRAM.

Un valore a True abilita l' esecuzione STEP.

Per ripartire da una PAUSA da STEP invocare il metodo **ExecuteProg(-1)**.

Altrimenti **StopProg()** per interrompere.

Ad ogni STEP di esecuzione del PartProgram viene invovato l' evento **PauseChanged**

4.14.6 Void StopProg()

Forza una STOP del PartProgram in esecuzione in corso
Se configurata la M di STOP, viene eseguita.

4.14.7 Int32 ReadLineaReal()

Legge la linea REALE in esecuzione sulo CN riferita al PART PROGRAM.

Questo metodo è analogo all' utilizzo delle' evento NLineaRealChanged.

Può comunque essere utilizzato quando il CN è in STOP dopo un allarme per ottenere la linea precisa di interruzione del PART PROGRAM.

4.15 Retrace

Questa classe gestisce tutta la logica della funzionalità di RETRACE di IsoNs.

Il RETRACE è un modo di funzionamento utile per alcune tipologie di macchine. Si tratta in pratica di scorrere il percorso utensile sul PEZZO con una sorta di simulazione, ma con assi in movimento. Lo scorrimento del percorso avviene in modalità JOG sia avanti che indietro.

Questo permette di scegliere come PUNTO DI RIPARTENZA qualsiasi tratto del PROFILO PEZZO anche se questo non è un punto di INIZIO o di FINE di un ELEMENTO. In pratica è possibile effettuare una RIPARTENZA da un punto qualsiasi di un ARCO o di una RETTA. Il fatto che gli assi siano effettivamente in movimento, permette di avere una visione reale sulla macchina del PUNTO DI RIPARTENZA.

Alla ripartenza da RETRACE può essere associata una funzione M preparatoria.

L'abilitazione della funzionalità RETRACE viene effettuata tramite il metodo InitRetrace()

4.15.1 Void ExecuteProg()

Avvia il PartProgram dal punto attuale dove si trovano gli assi sul profilo.

Se è stata configurata la M di GORETRACE, questa viene avviata. Viene invocato questo metodo quando l'operatore ha deciso il punto di ripartenza del profilo.

4.15.2 Void GoLine(Int32 Linea)

Sposta gli assi al blocco della linea indicata.

Serve per saltare parti di percorso utensile non interessate.

Linea Numero di linea del PartProgram Iso

4.15.3 Void InitRetrace()

Abilita la funzionalità RETRACE. Questo metodo deve essere invocato prima di utilizzare tutte le proprietà e gli altri metodi della classe RETRACE.

In pratica prepara una Lista di simulazione di tutto il PartProgram.

4.15.4 Void JogDown()

Sposta gli assi in senso NEGATIVO sul percorso PEZZO.

Per senso negativo si intende il senso verso l'inizio del PEZZO. Tutti i tratti del percorso utensile che rimangono dal punto attuale degli assi verso l'inizio del percorso vengono eseguiti in successione alla F impostata del tratto stesso.

Per fermare gli assi in un punto desiderato è sufficiente invocare il metodo **StopJog()**

Esempio C#

```
// evento button premuto
// avvia retrace in senso negativo
private void button1_MouseDown(object sender, MouseEventArgs e)
{
    IsoNs.Retrace.JogDown();
}
// evento button rilasciato
// ferma gli assi nel punto attuale
private void button1_MouseUp(object sender, MouseEventArgs e)
{
    IsoNs.Retrace.StopJog();
}
```

4.15.5 Void JogUp()

Sposta gli assi in senso POSITIVO sul percorso PEZZO.

Per senso positivo si intende il senso verso la fine del PEZZO. Tutti i tratti del percorso utensile che rimangono dal punto attuale degli assi verso la fine del percorso vengono eseguiti in successione alla F impostata del tratto stesso.

Per fermare gli assi in un punto desiderato e' sufficiente invocare il metodo **StopJog()**

Esempio C#

```
// evento button premuto
// avvia retrace in senso positivo
private void button1_MouseDown(object sender, MouseEventArgs e)
{
    IsoNs.Retrace.JogUp();
}
// evento button rilasciato
// ferma gli assi nel punto attuale
private void button1_MouseUp(object sender, MouseEventArgs e)
{
    IsoNs.Retrace.StopJog();
}
```

4.15.6 LineRetrace proprietà di tipo [Int32](#)- Read Only

Ritorna la linea del PartProgram attualmente in esecuzione nel Retrace

Aa ogni esecuzione di un Blocco viene comunque generato l' evento NlineaChanged()

4.15.7 Boolean GetPosAxisAtLine(out Int32 PosX, out Int32 Posy, Int32 Nline)

Ritorna in PosX e PosY la posizione di target degli assi contenuta nel blocco indicato da Nline

Linea **Numero di linea del PartProgram Iso**

4.16 Simula

Questa classe gestisce la simulazione grafica del PartProgram.

Per far si che questa possa essere utilizzata occorre allocare il file **Simulation.dll** che in pratica gestisce tutta la grafica della simulazione.

Questo si basa su un WindowsForm e viene gestito ne seguente modo:

Occorre aggiungere i seguenti riferimenti:

Simulation.dll

ComSymk.dll

```
Simulation.Simullso MySimula;
```

```
private void InitSimula()
{
    Double DimX, DimY, DimZ;
    System.Drawing.Size Sz = new Size();
    System.Drawing.Point Ptn = new Point();
    Sz.Width = // larghezza del FORM
    Sz.Height = // altezza del FORM
    Ptn.X = // Left FORM
    Ptn.Y = // Top FORM
    MySimula = new Simulation.Simullso(Sz, Ptn, IsoNs); // dimensioni del form e posizione
    // Se necessario e' possibile allocare un evento end simulazione
    // che viene generato quando il PartProgram in corso ha terminato la simulazione
    // dopo l' invocazione del metodo StartSimul()
    MySimula.MyPiano.EndSimul += new EventHandler<Simulation.SimulArgs>(EndSimul);
    DimX = IsoNs.Config.NsConfig.GetDimPianoX;
    DimY = IsoNs.Config.NsConfig.GetDimPianoY;
    DimZ = IsoNs.Config.NsConfig.GetDimPianoZ;
    MySimula.MyPiano.SetRealDimension(DimX, DimY, DimZ);
    MySimula.MyPiano.ResQuote = IsoNs.Config.NsConfig.GetValueParByName("RESQUOTE");
    // passa la classe al master
    IsoNs.MyMaster.SetClassSimula(MySimula); // Setta la classe al master
    MySimula.MyPiano.Owner = this; // FORM di appartenenza
}
```

NOTA IMPORTANTE

Per far si che la simulazione funzioni regolarmente e' necessario chiamare il metodo:

IsoNs.MyMaster.SetClassSimula(MySimula);

Questa passa il riferimento all classe MyMaster di IsoNs

4.16.1 Void StartSimul()

L' attuale PartProgram caricato viene eseguito in modalita' simulazione.

L' eventuale tracciato del percorso viene reso visibile nel FORM di simulazione caricato con Simulation.dll.

Da questo e' poi possibile utilizzare tutte le funzionalita' implementate.

Rifersi all documentazione di **Simulation.dll**

4.17 METODI ED EVENTI DI Simulalso.Piano

La classe MyPiano deriva da una classe Form e quindi eredita tutti gli eventi le proprietà e i metodi di questa. Oltre ai metodi e gli eventi ereditati ne vengono messi a disposizione altri che interessano la simulazione.

4.17.1 Void SetRealDimension(Double DimPianoX, Double DimPianoY, Double DimPianoZ)

Setta le dimensioni del piano di lavoro della Macchina (unità di misura mm)

Il piano di lavoro viene rappresentato da un rettangolo di colore verde. Questo serve per avere un raffronto tra dimensioni e posizione PEZZO rispetto al piano di lavoro

DimPianoX Dimensione in mm del piano nell' asse X

DimPianoY Dimensione in mm del piano nell' asse Y

DimPianoZ Dimensione in mm del piano nell' asse Z
(attualmente non è considerata)

4.17.2 ResQuote proprietà di tipo Int32- Write only

Setta la risoluzione delle quote

100 Centesimi di millimetro
1000 Millesimi di millimetro
10000 Decimillesimi di millimetro

...
..

4.17.3 EndSimul

Evento che si verifica quando la simulazione ha terminato il suo processo avviato dopo il metodo **StartSimul()**

4.17.4 Void SetQuota(Int32[] Val, Int32 Mask)

Muove la traccia di riferimento per simulazione lavorazione RealTime alle quote indicate. Generalmente questo metodo viene richiamato dall' evento di IsoNs QuoteTchanged

Esempio C#

```
private void QuoteTchanged(object sender, NsWork.QuoteArgs e)
{
    SimulaIso.Piano.SetQuota(e.RealValue,e.Mask);
}
```

4.17.5 SetDiaUt proprietà di tipo Double- Write only

Setta il diametro utensile per visualizzazione traccia in RealTime

Esempio C#

```
private void DiamChanged(object sender, NsWork.GenArgs e)
{
    SimulaIso.Piano.SetDiaUt = e.Value;
}
```

4.17.6 DrawNlinea proprietà di tipo Int32- Write only

Evidenzia il tratto riferito al numero di linea

Esempio C#

```
private void MarkLine(Int32 Nline)
{
    SimulaIso.Piano.DrawNlinea = Nline;
}
```

4.17.7 Void ClearListAll()

Ripulisce l' area di visualizzazione cancellando l' attuale disegno.
Questo metodo viene comunque chiamato automaticamente da **StartSimul()**

4.18 StatusCn

Questa classe contiene tutte le proprietà relative allo stato del CN.

Alcune di queste proprietà non sono aggiornate in tempo REALE, ma seguono il TIMER di sistema impostato nel file di configurazione **IsoNs.cfg** nel parametro **TIMERSCANNER**.

4.18.1 IsAbsRel proprietà di tipo **Boolean**- Read Only

True stato del CN e in movimentazione RELATIVA

False stato del CN e in movimentazione ASSOLUTA (dallo zero macchina impostato)

4.18.2 IsStatusEnableAxis proprietà di tipo **Boolean[]**- Read Only

Contiene un Array di tipo Boolean di dimensioni dipendenti dal numero di assi configurati. L' indice dell' asse rappresenta lo stato dell' abilitazione del driver di questo

True Asse abilitato

False Asse disabilitato

Esempio C#

```
// check assi abilitato
for(int n=0;n<IsoNs.Config.NsConfig.Nassi;n++)
{
    if(IsoNs.StatusCn.IsStatusEnableAxis[n]==false)
    {
        Label1.Text+="ASSE "+IsoNs.Config.NsConfig.NomiAssi[n]+" DISABLE.. ";
    }
    else
    {
        Label1.Text+="ASSE "+IsoNs.Config.NsConfig.NomiAssi[n]+" ENABLE.. ";
    }
}
```

4.18.3 IsStatusError proprietà di tipo **Boolean**- Read Only

Indica lo stato di errore del CN

True CN in Errore

False CN Ok

4.18.4 IsStatusHomeAxis proprietà di tipo **Boolean[]**- Read Only

Contiene un Array di tipo Boolean di dimensioni dipendenti dal numero di assi configurati. L' indice dell' asse rappresenta lo stato di homing

True Asse con homing effettuato

False Asse con homing non effettuato

Esempio C#

```
// check assi homing
for(int n=0;n<IsoNs.Config.NsConfig.Nassi;n++)
{
    if(IsoNs.StatusCn.IsHomingEnableAxis[n]==false)
    {
        Label1.Text+="ASSE "+IsoNs.Config.NsConfig.NomiAssi[n]+" NO HOMING. ";
    }
    else
    {
        Label1.Text+="ASSE "+IsoNs.Config.NsConfig.NomiAssi[n]+" HOMING OK. ";
    }
}
```

4.18.5 IsStatusM proprietà di tipo **Boolean**- Read Only
Indica lo stato di esecuzione funzioni M sul CN

True M in esecuzione
False M terminata

4.18.6 IsStatusPause proprietà di tipo **Boolean**- Read Only
Indica lo stato PAUSA del CN

True PAUSA in corso
False PAUSA terminata (stato del CN in RUN o STOP)

4.18.7 IsStatusPotVper proprietà di tipo **Boolean**- Read Only
Indica lo stato di abilitazione del potenziometro esterno di override

True Override esterno abilitato
False Override esterno disabilitato

4.18.8 IsStatusRun proprietà di tipo **Boolean**- Read Only
Indica lo stato di RUN/STOP del CN

True RUN in corso
False STOP in corso

4.18.9 IsStatusStepMode proprietà di tipo **Boolean**- Read Only
Indica lo stato di esecuzione del part program in modalità STEP MODE

True STEP MODE in corso
False Esecuzione normale

4.18.10 IsStatusAcq proprietà di tipo **Boolean**- Read Only
Indica lo stato dell' acquisizione da sensore in corso (da testare ovviamente dopo uno start Acq)

True Acq terminata
False Acq in corso

4.18.11 IsStatusMove proprietà di tipo **Boolean**- Read Only
Indica lo stato del movimento assi in corso

True Movimento in corso
False Assi fermi

4.19 VarIsoNs

Questa classe permette di accedere il lettura e scrittura alle variabili ISONS del part program.

Le variabili accessibili sono quelle generali di tipo \$.

La possibilità' di scrivere le variabili ha senso solamente se prima viene compilato un PartProgram questo poiché' solo dopo la compilazione si avranno i riferimenti precisi alle variabili.

Solamente alcune variabili di sistema rimangono inalterate come riferimento e so le variabili di scambio dati per funzioni M:

| Nome | Indirizzo in memoria |
|------------|----------------------|
| \$_PARAM_1 | 0 |
| \$_PARAM_2 | 1 |
| \$_PARAM_3 | 2 |
| \$_PARAM_4 | 3 |
| \$_PARAM_5 | 4 |
| ... | |

Il loro numero dipende dalla configurazione.

Queste variabili possono essere gestite anche prima della compilazione del PartProgram,

La possibilità' d scrivere le variabili del PartProgram permette di configurare ciclo di lavorazione in modo parametrico.

4.19.1 **Int32** GetAddrVar(**String** Nome)

Ritorna l' indirizzo in memoria IsoNs della variabile

Nome Nome della variabile escluso il carattere \$ iniziale

Ritorna

Int32 Indirizzo variabile
-1 variabile non trovata

4.19.2 **String** GetNomeVar(**Int32** Addr)

Ritorna il nome della variabile passata come indirizzo

Addr indirizzo in memoria IsoNs della variabile

Ritorna

String Nome della variabile escluso il carattere \$ iniziale
se **String=""** variabile non trovata

4.19.3 **Boolean** ReadVarIsoNs(**Int32** Addr, **Out Double** Val)

Overload

4.19.4 **Boolean** ReadVarIsoNs(**String** Nome,**Out Double** Val)

Ritorna il valore della variabile passata come indirizzo o come nome

Addr indirizzo in memoria IsoNs della variabile
Nome Nome della variabile escluso il carattere \$ iniziale
Val Valore della variabile in formato double

Ritorna

True variabile letta regolarmente
False variabile non trovata

4.19.5 Boolean WriteVarIsoNs(Int32 Addr, Double Val)

Overload

4.19.6 Boolean WriteVarIsoNs(String Nome, Double Val)

Scrive il valore nella variabile passata come indirizzo o come nome

| | |
|-------------|---|
| Addr | indirizzo in memoria IsoNs della variabile |
| Nome | Nome della variabile escluso il carattere \$ iniziale |
| Val | Valore da scrivere in formato double |

Ritorna

| | |
|-------------|--------------------------------|
| True | variabile scritta regolarmente |
|-------------|--------------------------------|

| | |
|--------------|-----------------------|
| False | variabile non trovata |
|--------------|-----------------------|

4.20.7 Void SetToPosition(Double[] Val, Int32 Index)

Setta lo ZeroPezzo all' indice indicato alle posizioni degli assi passati in Val

Val array di double caricato con la posizione di ZERO di ogni singolo asse
Index indice dello ZeroPezzo nell' ARRAY

Lo ZeroPezzo viene automaticamente ABILITATO

Genera eccezione se assi o indice non congruo

Esempio C#

```
private void SetZeroPezzo()
{
    Double[] ValZero = new Double[3]; // dimensiona array zero pezzo a tre assi
    ValZero[0] = 10.315; //carica le posizioni assi come zeropezzo
    ValZero[1] = 123.14;
    ValZero[2] = 0;
    IsoNs.ZeroPezzo.SetToPosition(ValZero, 0); //setta lo zero pezzo all' indice zero dell' array
}
```

4.20.8 Double[] GetZeroPosition(Int32 Index)

Ritorna la posizione di ZeroPezzo all' indice indicato attive in memoria.

Index indice dello ZeroPezzo nell' ARRAY

Ritorna

Array di Double contenente la posizione degli assi

Genera eccezione se indice non valido

4.20.9 Double[] GetZeroByFile(Int32 Index)

Ritorna la posizione di ZeroPezzo all' indice indicato salvate nel file di default ZERI.VAL

Index indice dello ZeroPezzo nell' ARRAY

Ritorna

Array di Double contenente la posizione degli assi

Genera eccezione se indice non valido

4.20.10 Void SetZeroSingleAxisToPosition(Double Val, Int32 Index, Int32 Axis)

Setta lo ZeroPezzo all' indice indicato dell' asse indicato al valore Val, lasciando inalterati gli zero degli altri assi

Val valore della posizione di ZERO dell' asse Axis
Index indice dello ZeroPezzo nell' ARRAY
Axis asse interessato
0 --> X 1 --> Y 2 --> Z ecc.

Lo ZeroPezzo viene automaticamente ABILITATO

Genera eccezione se assi o indice non congruo

4.20.11 **Void** SetZeroSingleAxisActualPosition(**Int32** Index, **Int32** Axis)

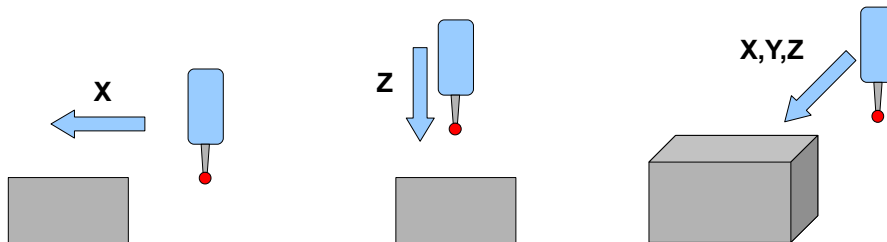
Setta lo ZeroPezzo all' indice indicato dell' asse indicato al valore attuale della posizione dell' asse, lasciando inalterati gli zero degli altri assi

Index indice dello ZeroPezzo nell' ARRAY
Axis asse interessato
0 --> X 1 --> Y 2 --> Z ecc.

Lo ZeroPezzo viene automaticamente ABILITATO
Genera eccezione se assi o indice non congruo

4.21 AcqSens

Questa classe gestisce le funzionalità di acquisizione assi da sensore. L' acquisizione da sensore prevede che gli assi vengano traslati ad una quota max ed entro questa traslazione venga rilevato il sensore. Gli assi possono essere traslati contemporaneamente è comunque possibile traslare un solo asse (cosa più logica) nella direzione di acquisizione del sensore.



4.21.1 Boolean StartAcqSens(Double[] QuoteAssi)

Avvia l' acq del sensore spostando gli assi alle quote indicate
L' acquisizione avvenuta viene rilevata tramite l' evento **AcqExecute** oppure **ErrorChanged** se errore acq.
È possibile testare il bit della StatusCn **IsStatusAcq**.

QuoteAssi Array con la posizione di Target in valore assoluto delle quote Assi
Devono essere passate tutte le quote assi.
Nel caso in cui uno o più assi non devono muoversi, inserire la posizione attuale
letta dal metodo **locs.ReadQuotaTeor**

Ritorna

True Acquisizione avviata

False Errore array quote assi non conforme al numero di assi presenti

ATTENZIONE

Prima di utilizzare il metodo StartAcqSens è necessario passare il CN in stato di RUN con il seguente codice:

```
NsWork.MyMaster.GetCommand.RunProg();
```

Alla fine della sessione di acquisizione occorre riportare il CN nello stato di STOP con il seguente codice:

```
NsWork.MyMaster.GetCommand.StopProg();
```

4.21.2 Double[] GetQuoteSens()

Ritorna le quote assi acquisite dal sensore.

Questo metodo va invocato dopo che l' acquisizione è terminata

Ritorna

Double[] Array contenente la quota degli assi dopo acq terminata

4.22 SaveVar

Questa classe gestisce i File di variabili salvate in memoria permanente. IsoNs permette di salvare il contenuto delle variabili in un file in modo che possa essere ricaricato successivamente. Questi file sono gestibili anche da PartProgram. Le variabili **PERMANENTI** sono tutte di tipo **DOUBLE** e vengono viste come una lista, pertanto ereditano tutti i metodi e proprietà dell' Interfaccia `Ilist<Double>`.

4.22.1 Boolean LoadFile(String Nome)

Carica in memoria il file indicato con Nome. Dopo l' invocazione di questo metodo è possibile utilizzare le variabile salvate.

Nome Nome del file da caricare

Ritorna

True File caricato regolarmente

False File non trovato

4.22.2 Void SaveFile(String Nome)

Salva il contenuto delle variabili PERMANENTI in un file.

Nome Nome del file da salvare

4.22.3 VarSave proprietà di tipo `IList<Double>`- Read/Write

Ritorna la lista delle variabili. Pertanto si possono utilizzare tutti i metodi dell' Interfaccia `Ilist<Double>`.

4.22.4 PathVarPerm proprietà di tipo `String` Read

Ritorna la path dove sono salvati i file delle variabili permanenti.

Indice generale

| | | |
|------|---|----|
| 1 | PREFAZIONE..... | 2 |
| 1.1 | Prima di iniziare..... | 2 |
| 2 | Eventi Disponibili..... | 3 |
| 2.1 | AbsRelChanged | 3 |
| 2.2 | DiamChanged | 3 |
| 2.3 | DigitalinputChanged | 3 |
| 2.4 | EnabledChanged | 4 |
| 2.5 | ErrorChanged | 4 |
| 2.6 | ExtRunChanged | 4 |
| 2.7 | ExtStopChanged | 4 |
| 2.8 | ExtPauseChanged | 5 |
| 2.9 | FeedChanged | 5 |
| 2.10 | HomeChanged | 5 |
| 2.11 | LenUtChanged | 5 |
| 2.12 | MexecuteChanged | 6 |
| 2.13 | MoveChanged | 6 |
| 2.14 | NlineaChanged | 6 |
| 2.15 | NlineaRealChanged | 6 |
| 2.16 | OnCloseCom | 6 |
| 2.17 | PauseChanged | 7 |
| 2.18 | PianoChanged | 7 |
| 2.19 | PotVperChanged | 7 |
| 2.20 | QuoteOffsetChanged | 7 |
| 2.21 | QuoteOffsetTestaChanged | 8 |
| 2.22 | QuoteRChanged | 8 |
| 2.23 | QuoteTchanged | 9 |
| 2.24 | QuoteZeriChanged | 9 |
| 2.25 | RemoveMark | 9 |
| 2.26 | ReqMarkLine | 9 |
| 2.27 | RunStopChanged | 10 |
| 2.28 | ScriptChanged | 10 |
| 2.29 | SpeedChanged | 10 |
| 2.30 | StepModeChanged | 10 |
| 2.31 | TabUtChanged | 11 |
| 2.32 | UserChanged | 11 |
| 2.33 | VperChanged | 11 |
| 2.34 | AcqExecute | 11 |
| 2.35 | MoltVolChanged | 12 |
| 2.36 | AxisJogChanged | 12 |
| 2.37 | StartImport..... | 12 |
| 2.38 | EndImport..... | 12 |
| 3 | Metodi e proprietà di NsWork..... | 13 |
| 3.1 | Void SetCn (Int32 IndexCn)..... | 13 |
| 3.2 | NsWork.IsoNs.ErrorCompiler[] CompileFileIso(String Text)..... | 13 |
| 3.3 | NsWork.IsoNs.ErrorCompiler[] CompileFromFile(String PathFile,out int32 TotalLine). 14 | |
| 3.4 | NsWork.IsoNs.ErrorCompiler[] CompileFromBlock(String PathFile,out int32 TotalLine)..... | 14 |
| 3.5 | EVENTI DI COMPILAZIONE..... | 15 |
| 3.6 | Void GoEvent....() | 15 |
| 3.7 | Boolean LoadCode()..... | 16 |

| | | |
|--------|--|----|
| 3.8 | Void MarkLine() | 16 |
| 3.9 | Void RemoveMarkLine() | 16 |
| 3.10 | Void ForceEventQuote() | 16 |
| 3.11 | Void EndSession() | 16 |
| 3.12 | GetMarker proprietà di tipo List<Compiler.MarkerCs> - Read | 16 |
| 4 | Classi di NsWork | 17 |
| 4.1 | BreakPoint | 17 |
| 4.1.1 | Int32[] BreakPoint.GetAllBreakPoint() | 17 |
| 4.1.2 | Void BreakPoint.InsertBreakPoint(Int32 LineNumber) | 17 |
| 4.1.3 | Boolean BreakPoint.IsBreakPoint(Int32 LineNumber) | 17 |
| 4.1.4 | Void BreakPoint.RemoveAllBreakPoint() | 17 |
| 4.1.5 | Void BreakPoint.RemoveBreakPoint(Int32 LineNumber) | 17 |
| 4.2 | ParTabUtHdCs | 18 |
| 4.2.1 | SelectTabUt proprietà di tipo Int32 - Read/Write | 18 |
| 4.2.2 | SelectHd proprietà di tipo Int32 - Read/Write | 18 |
| 4.2.3 | Double GetParTab(Double Index) | 18 |
| 4.2.4 | Double GetParHd(Double Index) | 18 |
| 4.2.5 | Void WriteParTab(Double Val,Int32 IndexTab,Int32 IndexPar) | 18 |
| 4.2.6 | Void SaveParTab(Int32 CnNumber) | 18 |
| 4.3 | CnError | 19 |
| 4.3.1 | String[] CnError.GetError() | 19 |
| 4.4 | CnIO | 20 |
| 4.4.1 | Int32 CnIO.ReadAnalogInput(Int32 Ninput) | 20 |
| 4.4.2 | Int32 CnIO.ReadAnalogOut(Int32 Nout) | 20 |
| 4.4.3 | Int32 CnIO.ReadChEncoder(Int32 Canale) | 20 |
| 4.4.4 | Int32 CnIO.ReadDigitalInput(Int32 Ninput) | 20 |
| 4.4.5 | Int32 CnIO.ReadDigitalOut(Int32 Nout) | 20 |
| 4.4.6 | Int32 CnIO.ReadGroupInputDig(Int32 Group) | 20 |
| 4.4.7 | Int32 CnIO.ReadGroupOutDig(Int32 Group) | 21 |
| 4.4.8 | Int32 CnIO.ReadQutaReal(Int32 Asse) | 21 |
| 4.4.9 | Int32 CnIO.ReadQutaTeor(Int32 Asse) | 21 |
| 4.4.10 | Void CnIO.SetResDigitalOut(Int32 Nout,Int32 Stato) | 21 |
| 4.4.11 | Void CnIO.WriteAnalogOut(Int32 Nout,Int32 Valore) | 21 |
| 4.4.12 | Byte[] CnIO.ReadByteCncMemory(Int32 Addr,Int32 Len) | 22 |
| 4.4.13 | Int32[] CnIO.ReadInt32CncMemory(Int32 Addr,Int32 Len) | 22 |
| 4.4.14 | Void CnIO.WriteByteCncMemory(Int32 Addr,Byte[] Data) | 22 |
| 4.4.15 | Void CnIO.WriteInt32CncMemory(Int32 Addr,Int32[] Data) | 22 |
| 4.5 | Config | 23 |
| 4.5.1 | Config.GetCncVersion proprietà di tipo String – ReadOnly | 23 |
| 4.5.2 | Config.NsConfig.AddressFixed proprietà di tipo Int32 - ReadOnly | 23 |
| 4.5.3 | Config.NsConfig.DebugMode proprietà di tipo Boolean - ReadOnly | 23 |
| 4.5.4 | Config.NsConfig.FullDebug proprietà di tipo Boolean - ReadOnly | 23 |
| 4.5.5 | Config.NsConfig.GetBaudRateRs232 proprietà di tipo Int32 - ReadOnly | 23 |
| 4.5.6 | String[] Config.NsConfig.GetCnError(Int32[] Allarm) | 23 |
| 4.5.7 | String Config.NsConfig.GetCodePause(Int32 Ncode) | 23 |
| 4.5.8 | Config.NsConfig.GetComType proprietà di tipo String - ReadOnly | 24 |
| 4.5.9 | Config.NsConfig.GetCpuType proprietà di tipo String - ReadOnly | 24 |
| 4.5.10 | Config.NsConfig.GetDefine proprietà di tipo List<String[]>- ReadOnly | 24 |
| 4.5.11 | String Config.NsConfig.GetDefineError(Int32 Nerror) | 24 |
| 4.5.12 | Config.NsConfig.GetDimPianoX proprietà di tipo Int32 - ReadOnly | 24 |
| 4.5.13 | Config.NsConfig.GetDimPianoY proprietà di tipo Int32 – ReadOnly | 24 |
| 4.5.14 | Config.NsConfig.GetDimPianoZ proprietà di tipo Int32 – ReadOnly | 24 |

| | | | |
|--------|---|---|----|
| 4.5.15 | Config.NsConfig.GetGottimizzate | proprietà di tipo Boolean- ReadOnly | 24 |
| 4.5.16 | Config.NsConfig.GetInsertNlinea | proprietà di tipo Boolean- ReadOnly | 24 |
| 4.5.17 | String Config.NsConfig.GetInternalError(Int32 Nerror) | | 25 |
| 4.5.18 | Config.NsConfig.GetMacroError | proprietà di tipo Int32- ReadOnly | 25 |
| 4.5.19 | Config.NsConfig.GetMacroGoBlock | proprietà di tipo Int32- ReadOnly | 25 |
| 4.5.20 | Config.NsConfig.GetMacroGoPause | proprietà di tipo Int32- ReadOnly | 25 |
| 4.5.21 | Config.NsConfig.GetMacroGoRetrace | proprietà di tipo Int32- ReadOnly | 25 |
| 4.5.22 | Config.NsConfig.GetMacroPause | proprietà di tipo Int32- ReadOnly | 25 |
| 4.5.23 | Config.NsConfig.GetMacroStop | proprietà di tipo Int32- ReadOnly | 25 |
| 4.5.24 | Config.NsConfig.GetNtab | proprietà di tipo Int32- ReadOnly | 25 |
| 4.5.25 | Config.NsConfig.GetNumComRs232 | proprietà di tipo Int32- ReadOnly | 25 |
| 4.5.26 | Config.NsConfig.GetNvarM | proprietà di tipo Int32- ReadOnly | 25 |
| 4.5.27 | Config.NsConfig.GetParMac | proprietà di tipo List<ComSynk.ParametriMacchina>- ReadOnly | 25 |
| 4.5.28 | Int32[] Config.NsConfig.GetParTab(Int32 Ntab) | | 26 |
| 4.5.29 | Config.NsConfig.GetPathIsoNs | proprietà di tipo String- ReadOnly | 26 |
| 4.5.30 | Config.NsConfig.GetRunErrComp | proprietà di tipo Boolean- ReadOnly | 26 |
| 4.5.31 | Config.NsConfig.GetSeqHoming | proprietà di tipo Int32[]- ReadOnly | 26 |
| 4.5.32 | Config.NsConfig.GetTimeOutRs232 | proprietà di tipo Int32- ReadOnly | 26 |
| 4.5.33 | Int32 Config.NsConfig.ValueParByName(String Name) | | 26 |
| 4.5.34 | Config.NsConfig.GruppiAllarm | proprietà di tipo Int32- ReadOnly | 26 |
| 4.5.35 | Config.NsConfig.GruppiInput | proprietà di tipo Int32- ReadOnly | 26 |
| 4.5.36 | Config.NsConfig.IpAddr | proprietà di tipo String- ReadOnly | 26 |
| 4.5.37 | Void Config.NsConfig.LoadConfigura() | | 27 |
| 4.5.38 | Config.NsConfig.MaxCodeMemory | proprietà di tipo Int32- ReadOnly | 27 |
| 4.5.39 | Config.NsConfig.MaxFeed | proprietà di tipo Int32- ReadOnly | 27 |
| 4.5.41 | Config.NsConfig.MaxSpeed | proprietà di tipo Int32- ReadOnly | 27 |
| 4.5.42 | Config.NsConfig.MaxVper | proprietà di tipo Int32- ReadOnly | 27 |
| 4.5.43 | Config.NsConfig.Nassi | proprietà di tipo Int32- ReadOnly | 27 |
| 4.5.44 | Config.NsConfig.NdecimalFeed | proprietà di tipo Int32- ReadOnly | 27 |
| 4.5.46 | Config.NsConfig.NdecimalQuote | proprietà di tipo Int32- ReadOnly | 27 |
| 4.5.47 | Config.NsConfig.NdecimalSpeed | proprietà di tipo Int32- ReadOnly | 27 |
| 4.5.48 | Config.NsConfig.NomiAssi | proprietà di tipo String[]- ReadOnly | 27 |
| 4.5.49 | Config.NsConfig.NunitaFeed | proprietà di tipo Int32- ReadOnly | 28 |
| 4.5.50 | Config.NsConfig.NunitaQuote | proprietà di tipo Int32- ReadOnly | 28 |
| 4.5.51 | Config.NsConfig.NunitaSpeed | proprietà di tipo Int32- ReadOnly | 28 |
| 4.5.52 | Config.NsConfig.Port | proprietà di tipo Int32- ReadOnly | 28 |
| 4.5.53 | Config.NsConfig.ReadInput | proprietà di tipo Int32[] - ReadOnly | 28 |
| 4.5.54 | Config.NsConfig.SogliaInsArchi | proprietà di tipo Double- ReadOnly | 28 |
| 4.5.55 | Config.NsConfig.TimeOut | proprietà di tipo Int32- ReadOnly | 28 |
| 4.5.56 | Config.NsConfig.TimeOutEnable | proprietà di tipo Int32- ReadOnly | 28 |
| 4.5.57 | Config.NsConfig.TimeOutHome | proprietà di tipo Int32- ReadOnly | 28 |
| 4.5.58 | Config.NsConfig.TimerScanner | proprietà di tipo Int32- ReadOnly | 29 |
| 4.5.59 | Config.NsConfig.UserGeneric | proprietà di tipo Int32- ReadOnly | 29 |
| 4.5.60 | Config.NsConfig.VeryAddressFixed | proprietà di tipo Int32 - ReadOnly | 29 |
| 4.6 | ExecuteMtoCn | | 30 |
| 4.6.1 | Boolean GoMToCn(Int32 CodeM, Int32[] Param) | | 30 |
| 4.6.2 | Int32 ReadParM(Int32 Npar) | | 30 |
| 4.6.3 | Boolean WriteParM(Int32 Npar,Int32 Valore) | | 30 |
| 4.6.4 | Void StopCnMacro() | | 30 |
| 4.7 | GestFonzMHM | | 31 |
| 4.7.1 | Boolean GenerateHM (Int32 Num, String CodeIso,out | | |

| | |
|--|----|
| NsWork.IsoNs.ErrorCompiler[] Err)..... | 31 |
| 4.7.2 Boolean GenerateM (Int32 Num, String CodeIso,out NsWork.IsoNs.ErrorCompiler[] Err)..... | 31 |
| 4.8 Homing..... | 32 |
| 4.8.1 Void EnableAxis(Int32 Naxis, Boolean Stato)..... | 32 |
| 4.8.2 GetNomiAssi proprietà di tipo String[]- ReadOnly..... | 32 |
| 4.8.3 GetSeqHoming proprietà di tipo Int32[]- ReadOnly..... | 32 |
| 4.8.4 Void StartHoming(Int32 Naxis)..... | 33 |
| 4.8.5 Void StopHoming()..... | 33 |
| 4.8.6 Int32 ReadSfasaTacca(Int32 Naxis)..... | 33 |
| 4.8.7 Void PresetAbsEnc(Int32 Naxis)..... | 33 |
| 4.9 JogAxis..... | 34 |
| 4.9.1 AbsRel proprietà di tipo Boolean- Read/Write..... | 34 |
| 4.9.2 ExtOverride proprietà di tipo Boolean- Read/Write..... | 34 |
| 4.9.3 FeedAxis proprietà di tipo Int32- Read/Write..... | 34 |
| 4.9.4 Void Jog(Boolen Direction)..... | 34 |
| 4.9.5 Void MoveAxis(Double Target, Int32 Axis)..... | 35 |
| 4.9.6 Void SetShiftAxis(Int32 Axis,Double ValShift)..... | 35 |
| 4.9.7 SelectAxisForJog proprietà di tipo Int32- Read/Write..... | 35 |
| 4.9.8 Void SelectAxisAndMolt(Int32 QualeAsse,Int32 MoltVol)..... | 35 |
| 4.9.9 Int32 ReadMolt()..... | 35 |
| 4.9.10 Void SopMove()..... | 35 |
| 4.10 MyMaster..... | 37 |
| 4.11 NewComp..... | 37 |
| 4.12 Param..... | 37 |
| 4.12.1 Void DownLoadPara()..... | 37 |
| 4.12.2 Boolean GetPar(Int32 Index, out String Nome,out String Descr,out String Gruppo,out Int32 Valore,out Int32 IndexCn)..... | 37 |
| 4.12.3 Boolean GetPar(String Nome, out Int32 Index,out String Descr,out String Gruppo,out Int32 Valore,out Int32 IndexCn)..... | 38 |
| 4.12.4 Gruppi proprietà di tipo String[]- Read Only..... | 38 |
| 4.12.5 Npar proprietà di tipo Int32- Read Only..... | 38 |
| 4.12.6 Boolean RipristinaBak()..... | 38 |
| 4.12.7 Void SaveCfg()..... | 38 |
| 4.12.8 Void SendAllPara()..... | 38 |
| 4.12.9 Void UpdatePara()..... | 38 |
| 4.12.10 Boolean WritePara(Int32 Index, Int32 Value, Boolean WriteCn)..... | 39 |
| 4.12.11 Boolean WritePara(String Name, Int32 Value, Boolean WriteCn)..... | 39 |
| 4.12.12 SetVisuaQuoteReal proprietà di tipo NsWork.ParamCs.VisType- Read/Write..... | 39 |
| 4.13 ParamUserGeneric..... | 40 |
| 4.13.1 Void WriteParamUser(Int32 Addr,Int32 Value)..... | 40 |
| 4.14 ProgramRun..... | 41 |
| 4.14.1 Boolean ExecuteScript(String Code, out NsWork.IsoNs.ErrorCompiler[] Err)..... | 41 |
| 4.14.2 Void ExecuteProg(Int32 Nlinea)..... | 41 |
| 4.14.3 Void ExecuteProgFromMarker(Int32[] AddrMarker,Double[] ValMarker)..... | 42 |
| 4.14.4 Void PauseProg()..... | 43 |
| 4.14.5 StepMode proprietà di tipo Boolean- Read/Write..... | 43 |
| 4.14.6 Void StopProg()..... | 43 |
| 4.14.7 Int32 ReadLineaReal()..... | 43 |
| 4.15 Retrace..... | 44 |
| 4.15.1 Void ExecuteProg()..... | 44 |
| 4.15.2 Void GoLine(Int32 Linea)..... | 44 |

| | | |
|---------|---|----|
| 4.15.3 | Void InitRetrace() | 44 |
| 4.15.4 | Void JogDown() | 44 |
| 4.15.5 | Void JogUp() | 45 |
| 4.15.6 | LineRetrace proprietà di tipo Int32- Read Only | 45 |
| 4.15.7 | Boolean GetPosAxisAtLine(out Int32 PosX, out Int32 Posy, Int32 Nline) | 45 |
| 4.16 | Simula | 46 |
| 4.16.1 | Void StartSimul() | 46 |
| 4.17 | METODI ED EVENTI DI SimulaIso.Piano | 47 |
| 4.17.1 | Void SetRealDimension(Double DimPianoX,Double DimPianoY,Double DimPianoZ) | 47 |
| 4.17.2 | ResQuote proprietà di tipo Int32- Write only | 47 |
| 4.17.3 | EndSimul | 47 |
| 4.17.4 | Void SetQuota(Int32[] Val, Int32 Mask | 47 |
| 4.17.5 | SetDiaUt proprietà di tipo Double- Write only | 47 |
| 4.17.6 | DrawNlinea proprietà di tipo Int32- Write only | 48 |
| 4.17.7 | Void ClearListAll() | 48 |
| 4.18 | StatusCn | 49 |
| 4.18.1 | IsAbsRel proprietà di tipo Boolean- Read Only | 49 |
| 4.18.2 | IsStatusEnableAxis proprietà di tipo Boolean[]- Read Only | 49 |
| 4.18.3 | IsStatusError proprietà di tipo Boolean- Read Only | 49 |
| 4.18.4 | IsStatusHomeAxis proprietà di tipo Boolean[]- Read Only | 49 |
| 4.18.5 | IsStatusM proprietà di tipo Boolean- Read Only | 50 |
| 4.18.6 | IsStatusPause proprietà di tipo Boolean- Read Only | 50 |
| 4.18.7 | IsStatusPotVper proprietà di tipo Boolean- Read Only | 50 |
| 4.18.8 | IsStatusRun proprietà di tipo Boolean- Read Only | 50 |
| 4.18.9 | IsStatusStepMode proprietà di tipo Boolean- Read Only | 50 |
| 4.18.10 | IsStatusAcq proprietà di tipo Boolean- Read Only | 50 |
| 4.18.11 | IsStatusMove proprietà di tipo Boolean- Read Only | 50 |
| 4.19 | VarIsoNs | 51 |
| 4.19.1 | Int32 GetAddrVar(String Nome) | 51 |
| 4.19.2 | String GetNomeVar(Int32 Addr) | 51 |
| 4.19.3 | Boolean ReadVarIsoNs(Int32 Addr, Out Double Val) | 51 |
| 4.19.4 | Boolean ReadVarIsoNs(String Nome,Out Double Val) | 51 |
| 4.19.5 | Boolean WriteVarIsoNs(Int32 Addr,Double Val) | 52 |
| 4.19.6 | Boolean WriteVarIsoNs(String Nome,Double Val) | 52 |
| 4.20 | ZeroPezzo | 53 |
| 4.20.1 | Void Disable(Int32 Index) | 53 |
| 4.20.2 | IndexZero proprietà di tipo Int32- Read/Write | 53 |
| 4.20.3 | SospendiZeri proprietà di tipo Boolean- Read/Write | 53 |
| 4.20.4 | SospendiOffset proprietà di tipo Boolean- Read/Write | 53 |
| 4.20.5 | SospendiOffsetTeste proprietà di tipo Boolean- Read/Write | 53 |
| 4.20.6 | Void SetActualPosition(Int32 Index) | 53 |
| 4.20.7 | Void SetToPosition(Dounle[] Val, Int32 Index) | 54 |
| 4.20.8 | Dounle[] GetZeroPosition(Int32 Index) | 54 |
| 4.20.9 | Dounle[] GetZeroByFile(Int32 Index) | 54 |
| 4.20.10 | Void SetZeroSingleAxisToPosition(Dounle Val, Int32 Index, Int32 Axis) | 54 |
| 4.20.11 | Void SetZeroSingleAxisActualPosition(Int32 Index, Int32 Axis) | 55 |
| 4.21 | AcqSens | 56 |
| 4.21.1 | Boolean StartAcqSens(Double[] QuoteAssi) | 56 |
| 4.21.2 | Double[] GetQuoteSens() | 56 |
| 4.22 | SaveVar | 57 |
| 4.22.1 | Boolean LoadFile(String Nome) | 57 |

| | | |
|--------|--|----|
| 4.22.2 | Void SaveFile(String Nome)..... | 57 |
| 4.22.3 | VarSave proprietà di tipo IList<Double>- Read/Write..... | 57 |
| 4.22.4 | PathVarPerm proprietà di tipo String Read..... | 57 |