IsoUs – Ultimate Step Framework for .NET

www.promax.it

PROMAX Motion & Control

Rev. 1.3.0 © Promax srl

1 PREFAZIONE

This manual, decsribes the IsoUs Framework for .NET application.
The framewotk is contained in the **UsWork.dll**.

1.1 Before Start

The use of UsWork.DLL provides that there is the configuration file **"IsoUs.cfg"** already installed in the same folder.
The configuration file contains all the informations necessary to component for the proper functioning and adaptation to the CNC in use.
In the same folder UsWork.DLL must be the following DLL:

Compiler.DLL
ComSynk.DLL

These are directly allocated by the component itself and therefore only require their presence.
The component is loaded in the development environment like all other components of the framework running the standard usage.

First you must assign the address of the property SetIsoUsCnC CN (usually 0)

```
UsWork.IsoUs MyUsIso = new UsWork.IsoUs();

    // Run Application
    private void StartIsoUs()
    {
        SetIsoUsCnC(CnCIndex, StartupPath, "IsoUs.cfg");
    }
```

2 Events

2.1 AbsRelChanged

Event occurs to change the state of NC by absolute or relative. Generated as a result of an instruction manual functions G90 or G91.
False indicate Absolute Motion
True indicate Relative Motion

Example c#:

```
private void AbsRelChanged(object sender, UsWork.BoolArgs e)
{
  if (e.Value == false)
    … Absolute
  else
    … Relative
}
```

2.2 AcquisitionExecuteChanged

Event occurs to change the bit of a statusword **"probe acquisition"**.
Boolean value that indicates the status of acquisition

True	acquisition terminated
False	acquisition in RUN

Example c#:

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

2.3 AllAxesReadyChanged

Event occurs to change the Axes Status (homing or Enable)
Boolean value that indicates the axes sttus

True	All Axes Reday (Homing e Enable)
False	One o more Axis not ready (Homing e Enable)

2.4 AxesDemandValueChanged

Event occurs to change the demand position of Axes

FormatRealValue	contains an array of string formatted values of the absolute reference positions axes to machine zero
FormatSpaceErr	contains an array of string formatted values of the axes of space errors (difference between the theoretical position and current position)
Format	Contains an array of string formatted values of the axes positions These are referred to the axis and Zero Offset
RealValue	contains an array of Int32 values of the coordinates axes to machine zero absolute reference
Value	contains an array of Int32 values of the positions These are referred to the axis and Zero Offset
Mask	Int32 Bit mapped that indicates which axis has changed positions Bit 0 Axis X - Bit 1 Axis Y etc. .. The bit is not enabled, the event did not generate

Example c#:

```
private void AxesDemandValueChanged(object sender, UsWork.AxesValueArg e)
{
    Int32 P = 1;
     for (int n = 0; n < e.FormatValue.Length; n++)
    {
            if ((Mask & P) == P)    // value changed
            {
                    MyQuoteAbs[n].Text = FormatRealValue[n];
                    MyQuoteRel[n].Text = FormatValue[n];
            }
     P <<= 1;
    }
}
```

2.5 *AxesMoveChanged*

Event occurs to change the Axes movement

True	Axis movement
False	Axis stop

Example c#:

```
private void MoveChanged(object sender, NsWork.BoolArgs e)
{
  if (e.Value == false)
    axis stop
  else
   axis movement
}
```

2.6 *AxesOffsetValueChanged*

Event occurs to change the offset axes following a G93 statement

Format	containing a value formatted string
Index	index of the axis
Value	containing the integer value of the offset axis

Example c#:

```
private void AxesOffsetValueChanged(object sender, UsWork.OffsetArgs e)
{
  labelval.Text = e.Format;
  Int32 Value = e.Value;
  Double ValZero = e.Value;
  ValZero /= MyUsIso.UsAxesValue.AxesResolution;
}
```

2.7 *AxesOriginChanged*

Event occurs to change the axis zeros following a G94 statement

Format	String position formatted value
Index	Axis index
Value	Int32 offset axis

Example c#:

```
private void AxesOriginChanged (object sender, UsWork.OffsetArgs e)
{
  labelval.Text = e.Format;
  Int32 Value = e.Value;
  Double ValZero = e.Value;
  ValZero /= MyUsIso.UsAxesValue.AxesResolution;
}
```

2.8 AxesDemandValueChanged

Event occurs to change the real position of Axes

FormatRealValue	contains an array of string formatted values of the absolute reference positions axes to machine zero
FormatSpaceErr	contains an array of string formatted values of the axes of space errors (difference between the theoretical position and current position)
Format	contains an array of string formatted values of the axes positions These are referred to the axis and Zero Offset
RealValue	contains an array of Int32 values of the coordinates axes to machine zero absolute reference
Value	contains an array of Int32 values of the positions These are referred to the axis and Zero Offset
Mask	Int32 Bit mapped that indicates which axis has changed positions Bit 0 Axis X - Bit 1 Axis Y etc. .. The bit is not enabled, the event did not generate

2.9 AxisEnabledChanged

Event occurs to change the enabled status of the driver of an axis
Value contains the status

True	Axis enabled
False	Axis disabled

IndexAxis contains the index of axis (0-1-2) eg X, Y, Z

Example c#:

```
private void AxisEnabledChanged (object sender, UsWork.HomingEnableArgs e)
{
  if (e.Value == false)
     e.IndexAxis...      //Disable
  else
     e.IndexAxis...      // Enable
}
```

2.10 AxisHomeChanged

Event that occurs when an axis has finished homing function
Value contains the status of Home

True	Axis homing OK
False	Axis not homing

IndexAxis contains the index of the axis (0-1-2) eg X, Y, Z

Example c#:

```
private void AxisHomeChanged (object sender, UsWork.HomingEnableArgs e)
{
  if (e.Value == false)
     e.IndexAxis...      // no homing
  else
     e.IndexAxis...      // homing
}
```

2.11 *PositDemandValueChanged*

Event occurs to change the real position of POSITIONER Axes

Value	contains an array of Int32 values of the positions
Mask	Int32 Bit mapped that indicates which axis has changed positions Bit 0 Axis X - Bit 1 Axis Y etc. .. The bit is not enabled, the event did not generate

Example c#:

```
private void PositDemandValueChanged(object sender, UsWork.PositiValueArg e)
{
   Int32 P = 1;
    for (int n = 0; n < e.Value.Length; n++)
   {
          if ((Mask & P) == P)   // Changed
          {
                 MyQuoteAbs[n].Text = Value[n];
          }
    P <<= 1;
   }
}
```

2.12 *PosHomeChanged*

Event that occurs when an POSITIONER has finished homing function

Value contains the status of Home

True	Axis homing OK
False	Axis not homing

PosIndex contains the index of the POSITIONER (0-1-2)

Example c#:

```
private void PosHomeChanged (object sender, UsWork.  PosBoolArgs e)
{
  if (e.Value == false)
     e.PosIndex...        // no homing
  else
      e. PosIndex...      // homing
}
```

2.13 *PosEnabledChanged*

Event occurs to change the enabled status of the driver of an POSITIONER

Value contains the status

True	Axis enabled
False	Axis disabled

PosIndex contains the index of POSITIONER (0-1-2)

Esempio c#:

```
private void PosEnabledChanged (object sender, UsWork.  PosHomingEnableArgs e)
{
  if (e.Value == false)
     e.PosIndex...        // disabled
  else
      e. PosIndex...      // enabled
}
```

2.14 PosMoveChanged

Event occurs to change the POSITIONER movement

True movement

False stop

Example c#:

```
private void PosMoveChanged(object sender, UsWork.PosBoolArgs e)
{
   if (e.Value == false)
     e.PosIndex // Stop
   else
    e.PosIndex  // Move
}
```

2.15 PosAlarmChanged

Event occurs during ALLARMA POSITIONER

Value true ALLARM

Example c#:

```
private void PosAlarmChanged(object sender, UsWork.PosBoolArgs e)
{
   if (e.Value == false)
     e.PosIndex // Restored Alarm
   else
    e.PosIndex  // Alarm
}
```

2.16 DigitalInputChanged

Event occurs to change the status of a digital input.
The digital inputs enabled the generation of events must first be configured in the file **IsoUs.cfg** in the following mode:
<EventsData UserGeneric="" DigitalInputs=" inputn,inputn1,inputn2 etc " />
Where Inputn1,Inputn2 ecc. are the NC real digital inputs
Ex:
<EventsData UserGeneric="" DigitalInputs=" 1,5,32 " />

Example c#:

```
private void DigitalInputChanged(object sender, UsWork.DigitalInputArgs  e)
{
   for(int n=0;n<e.DigitalInput.Length;n++)
   {
      // e.DigitalInput is a integer vector that contains the digital inputs changed
      // ex: e.DigitalInput[0]=digital input 1 changed
      //  e.DigitalStato is a boolean vector that  contains the state of relative input
      // e:x e.DigitalStato[0]=false Input 1  false
      // test input 1
      if(e.DigitalInput[n]==1)
      {
         if(e.DigitalStato[n]==true)
         {
            if(IsoNs .IsStatusRun==true)     // if run
               IsoNs.ProgramRun.PauseProg();
         }
      }
   }
}
```

2.17 EnableExtOverrideChanged

Event occurs to change the enabled status of the external potentiometer override

True	external override enbled
False	external override disabled

Example c#:

```
private void EnableExtOverrideChanged (object sender, UsWork.BoolArgs e)
{
   if (e.Value == false)
      .....Disable
   else
     ..... Enable
}
```

2.18 EndImportChanged

Event occurs when the last IMPORT file is terminated

e.FileName	NULL
e.FileImportPath	Path file that has terminated the execution

2.19 ExtPauseRequestChanged

Event that occurs at the change of external input **PAUSE** program
The input is defined in the CN application VTB
True request from external **PAUSE**

Example c#:

```
private void ExtPauseRequestChanged (object sender, UsWork.BoolArgs e)
{
  if (e.Value == true)
    MyUsIso.UsGcodeRun.PauseGcode();
}
```

2.20 ExtRunRequestChanged

Event that occurs at the change of external input **RUN** program
The input is defined in the CN application VTB
True request from external **RUN**

Example c#:

```
private void ExtRunRequestChanged(object sender, UsWork.BoolArgs e)
{
  if (e.Value == true)
    MyUsIso.UsGcodeRun.ExecuteGcode(LineStart);      // Run Prog
}
```

2.21 ExtStopRequestChanged

Event that occurs at the change of external input **STOP** program
The input is defined in the CN application VTB
True request from external **STOP**

Example c#:

```
private void ExtStopRequestChanged(object sender, UsWork.BoolArgs e)
{
  if (e.Value == true)
    MyUsIso.UsGcodeRun.StopGcode();
}
```

2.22 FeedAxesChanged

Event occurs to change the set F
The event is generated by **F** instructions or manual functions
Contains an integer value of **F**
Format contains a string formatted to units of measurement set

Example c#:

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

2.23 RealFeedAxesChanged

Event occurs to change the Real Axes Feed calculato in the CNC
Must be enabled the parameter ENABLE_RFEED
Contains an integer value of **F**
Format contains a string formatted to units of measurement set

Esempio c#:

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

2.24 HeadOffsetValueChanged

Event occurs to change the offset head following a **Hn** command

Format	containing a value formatted string
Index	index of the axis
Value	containing the integer value of the offset axis

Example c#:

```
private void HeadOffsetValueChanged (object sender, UsWork.OffsetArgs e)
{
   labelval.Text = e.Format;
   Int32 Value = e.Value;
   Double ValZero = e.Value;
   ValZero /= MyUsIso.UsAxesValue.AxesResolution;;
}
```

2.25 MachineParametersUpdate

Evento occurs when one ore more machine parameters are changed by browser.

2.26 MulHandWheelChanged

Event occurs to change the multiplier value software for electronic handwheel. This event is generated even if the multiplier varies from application through VTB switch.
Value Int32 Contains the multiplier value

Example c#:

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

2.27 M_CnC_ExecuteChanged

Event that occurs at the end of a **M** function internal to the NC
True M in RUN
False M terminated

Example c#:

```
private void M_CnC_ExecuteChanged (object sender, UsWork.BoolArgs e)
{
   if(e.Value==true)
      // M START
   else
      // M terminated
}
```

2.28 OnCloseComRequest

Event occurs to end communication to NC

2.29 OverrideValueChanged

Event occurs to change the value of the Override Percentage FEED velocity axis.

Value	Int32 analog input value (0-1023)
PercValue	Percentage

Example c#:

```
private void OverrideValueChanged (object sender, UsWork.OverrideArgs e)
{
   LedOverride.Level = e.Value;
   LblOw.Text = e.PercValue + " %";
}
```

2.30 PauseChanged

Event occurs to change the pause state of NC
True NC in pause
False resume from pause

Example c#:

```
private void PauseChanged(object sender, UsWork.PauseArgs e)
{
   if (e.Value == false)
       // Resume
   else
        // PAUSE
}
```

2.31 RemoveMarkLineChanged

Event application required by simulation that enables an application to remove the marker line ISO. This event has a specific use is enabled only if the simulation with the file **Simulation.dll**. This file may take to the main interface to remove the marker line after an event **RequestMarkLineChanged**

2.32 RequestMarkLineChanged

Event application required by simulation that enables an application to mark a line in the Gcode ISO This event has a specific use is enabled only if the simulation with the file **Simulation.dll**. This file may take to mark a main interface
ISO line to highlight it

2.33 RunStopChanged

Event that occurs at the change of state of the NC STOP RUN
True NC in RUN
False NC in STOP

Example c#:

```
private void RunStopChanged(object sender, UsWork.BoolArgs e)
{
   ProgInRun = e.Value;
   if (e.Value == false)
      // from RUN to STOP
   else
      // from STOP to RUN
}
```

2.34 RunStopChangedWithType

Event that occurs at the change of state of the NC STOP RUN with RUN type
True NC in RUN
False NC in STOP

Example c#:

```
private void RunStopChangedWithType (object sender, UsWork.TypeRunArgs e)
{
      // TypeRun=0 Normal Run
      // TypeRun=1 Preview Run
      // TypeRun=2 Resume from Block Run
      // TypeRun=3 Retrace Run
      // TypeRun=4 Execution Time Calc Run
}
```

2.35 ScriptChanged

Event that occurs at the RUN / STOP ISONS script code that is invoked when the method IsoNs.ProgramRun.ExecuteScript

True Script RUN

False Script terminated

Example c#:

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

2.36 SelectAxisJogChanged

Event occurs to change the value of the selected axis to jog.

e.value axis selected

0 → Axis X	**1 → Axis Y**	**2 → Axis Z**
3 → Axis A	**4 → Axis B**	**5 → Axis C**
6 → Axis U	**7 → Axis V**	**8 → Axis W**

Example c#:

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

2.37 SpeedSpindleChanged

Event occurs to change the spindle speed in response to an instruction **S** from Gcode

Value Int32 **S** value

Format String formatted value

PercValue Double percentage of maxvalue (maxvaluesetted in isous.cfg)

Example c#:

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

2.38 StartImportChanged

Event occurs when IMPORT instruction is executed

e.FileName Text of imported file.

e.FileImportPath File of imported file

2.39 StepModeChanged

Event occurs to change the state of NC by way of execution STEP by STEP (line by line) to the continuous play mode.
True Step Mode
False Normal mode

Example c#:

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

2.40 TabUtChanged

Event occurs to change the selected value of the tool table.
ISO function **Tn**
Value Int32 **T** number selected

Example c#:

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

2.41 ToolDiameterChanged

Event occurs to change the settings of the tool diameter.
Created in response to an instruction of the **D** Gcode. The value is contained in an integer and represents the thousandths of a millimeter in diameter tool.

Example c#:

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

2.42 ToolLengthChanged

Event occurs to change the tool length set by the Gcode
Value contains the value in thousandths of a millimeter of the tool length

Example c#:

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

2.43 UsFatalError

If this event is allocated, IsoUs does'not manages the fatal Errors by notify the external application can get the events from Fatal Errors

```
void MyUsIso_UsFatalError(object sender, UsWork.UsFatalErr e)
{
   MessageBox.Show("ERROR CN : " + e.UsCnNumber.ToString() + "\n" + e.UsErrorMessage);
}
```

2.44 UsNotifyChanged

System Notify

Type:

CNC ALARM	Alarm on CNC
CNC WARNING	Warning on CN
DEFINE ERROR	Defined Error by user (ERROR Gcode Function)
GCODE ERROR	Error in Gcode
INFORMAZIONI	Info

Example c#:

```
private void MyUsIso_UsNotifyChanged(object sender, UsWork.UsNotify e)
{

          switch(e.NotifyType)
          {
            case UsWork.IsoUs.UsNotifyType.UsCncAlarm:
              // CNC in Alarm
              MessageBox.Show("CNC ALARM : " + e.NotifyDescription);
              break;
            case UsWork.IsoUs.UsNotifyType.UsCncWarning:
              // CNC in Warning
              MessageBox.Show("CNC WARNING : " + e.NotifyDescription);
              break;
            case UsWork.IsoUs.UsNotifyType.UsGcodeDefineError:
              // Error define by User
              MessageBox.Show("CNC ERROR DEFINE BY USER : " + e.NotifyDescription);
              break;
            case UsWork.IsoUs.UsNotifyType.UsGcodeError:
              // Error in Gcode
              MessageBox.Show("ERROR IN GCODE : " + e.NotifyDescription);
              break;
            case UsWork.IsoUs.UsNotifyType.UsInfo:
              // Information
              MessageBox.Show("INFO : " + e.NotifyDescription);
              break;
          }

  }
```

2.45 VariableUserChanged

Event that occurs at the change of a variable USER GENERIC IsoUs.cfg configured.
GENERIC USER variables can be used for data exchange between NC and PC (since these are manageable by applying the CN VTB)
Value Int32 Array that contains the value of USER changed
Mask Bit mapped indicate which USER is changed
Bit 0 User configured 1
Bit 1 User configured 2 etc.
Only those with bit 1 are changed

Example c#:

```
private void VariableUserChanged(object sender, UsWork.UserGeneric e)
{
   Int32 P=1;
   for(int n=0;n<e.Value.Length;n++) // check variable changed
   {
      if((e.Mask & P)==P) // variable changed
         LblUser.Text = e.Value[n].ToString();
      P <<= 1;    // shift bit
   }
}
```

2.46 WorkLineDemandChanged

Event occurs to change the number of demand lines running
Value contains an integer number of demand line running

Example c#:

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

2.47 WorkLineRealChanged

Event occurs to change line number REAL running on CN
This line number may differ from the demand.
Value contains an integer number of CURRENT line running

Example c#:

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

2.48 WorkPlaneSetChanged

Event occurs to change the work plan as a result of the NC **G17 G18 G19 G70** instructions
WorkPlaneSet contains an array of string with the names of the two axes

Example c#:

```
private void PianoChanged(object sender, UsWork.WorkPlaneArgs e)
{
   LblPiano1.Text = e.WorkPlaneSet0];
   LblPiano2.Text = e.WorkPlaneSet0[1];
}
```

2.49 UsCompiler.CodeLoaded

Event occurs when the compiled code is loaded in the IsoUs memory
Now is ready to work

2.50 G43Changed

Event occurs when the G43 changes its state

Value	**True** G43 Enabled
	False G43 Disabled

2.51 Us2ndLimitsChanged

Event occurs when the 2nd limits are changed
Value
True 2nd Limits Activated
False 2nd Limits Deactivated

2.52 UsDebuInfo

Event occurs when is used by Gcode:
DEBUG_INFO "TEXT$" $VAR with Last char of Text **$**

3 Methods and Properties of UsWork

3.1 *Double AxisValueToDoubleUs (Int32 AxisIntValue)*

Return a double value of an Integer Axis Value.
The return value considers the Axes Resolution

3.2 *Void PutNotify(String NotifyText, bool PutInPanel=false)*

Sends a notify to Notify Manager.
Follows an event UsNotifyChanged if PutInPanel=True
The notify is also put in the IsoUs_x.log file

3.3 *Void PutNotify(UsNotifyType NotifyType,String NotifyText, bool PutInPanel=false)*

Sends a notify to Notify Manager.
NotifyType defines the Type:

UsCncAlarm
UsCncWarning
UsGcodeError
UsGcodeDefineError
UsInfo

Follows an event UsNotifyChanged if PutInPanel=True
The notify is also put in the IsoUs_x.log file

3.4 *Void EndSession()*

Close all communications

3.5 *Void ForceEventAxesValue()*

Force the Events
AxesDemandValueChanged
AxesRealValueChanged
AxesOriginsChanged
AxesOffsetChanged

3.6 *Void SetIsoUsCnC (Int32 IndexCn, String CfgName)*

Init the CnC session

IndexCn	Index of CNC from 0 to 7 (multiprocess)
CfgName	Name of configuration file (normally **“IsoUs.cfg”)**

Before to call **SetIsoUsCnC** the keys that indicate the location and configuration to be loaded must be registered
The CNC can’t be connected to PC. This means the the IsoUs is in DEMO MODE.
Read **IsoUsModeRun**. If **False** the CNC is in DEMO MODE

3.7 *Bool IsoUsModeRun*

Property Read
True CNC in NORMAL MODE
False CNC in DEMO MODE

3.8 *Int32[] GetIndexCnC*

Property Read
Get the CnC Index setted by **SetIsoUsCnC**

3.9 Int32[] *WorkPlaneSet*

Property Read Write
Work plane setted
Array of Int32[2:
Int32[0] Index of the first Axis of Work Plane
Int32[1] Index of the second Axis of Work Plane

```
Int32[] _Plane = new Int32[2];
_Plane[0] = 0; //X
_Plane[1] = 2; //Z
MyUsIso.WorkPlaneSet = _Plane;
```

3.10 String *GetNameConfigIsoUs*

Property Read
Return the name of configuration file

3.11 String *GetPathIsoUs*

Property Read
Return the Path of configuration file

4 Classi di UsWork

4.1 MyMaster

Do not use

4.2 *UsAxesHomingEnable*

It contains all the methods that manage the Property **HOMING** and **ENABLE** of the axes

4.2.1 *bool* AllAxesRedy

Property Read

True	All Axes Reday – Homing and Enable performed
False	All Axes not ready

4.2.2 *bool* NoCheckAllAxesRedy

Property Read/Write

True AllAxesRedy return always True

4.2.3 Int32[] GetAxesHomingSequence

Property Read

Returns the axes sequence homing configured

Ex:

AXES CONFIGURED X,Y,Z
Sequence configured 2,1 (0 not homing)
Return GetSeqHoming Int32 arrayi:
Int32[0]=2
Int32[1]=1
Int32[0]=0

The first axis is Z
The second axis is Y
The third axis is X

4.2.4 Void EnableAxis(Int32 AxisIndex, bool State)

Enable/Disable Axis

AxisIndex	Index of axis
State	**True** Enable
	False Disable

The number of axis refers to the index of axes configured (0=X – 1=Y 2=Z etc.)

At this stage you can manage the parameter **TIMEOUTENABLE**

4.2.5 Void PresetAbsoluteEncoder(Int32 AxisIndex)

Method that executes the preset ZERO for multi-turn absolute encoder type.

This method sets the current position of the CN as ZERO. The method should be used when you want to fix the ABSOLUTE ZERO for the first time.

AxisIndex	Index of axis

4.2.6 Int32 ReadEncoderIndexShift(Int32 AxisIndex)

Method that reads the phase shift in the pulse encoder zero mark in relation to micro-homing.

The value is updated after searching the zero axis.

This method is valid only for boards that are connected encoder with zero mark on the CN

AxisIndex	Index of axis

4.2.7 bool ResetAxesHoming(bool[] Axes)

Reset the Homing

Axes	Array of bool. The position with **True** indicate, the reset bit of axis

4.2.8 Void **StartHomingAxis**(Int32 **AxisIndex**)

Run start axis homing procedure
The number of axis refers to the index of axes configured (0=X – 1=Y 2=Z etc.)
At this stage you can manage the parameter TIMEOUTHOME

AxisIndex	Index of Axis

4.2.9 Void **StopHomingAxis**()

Stop homing sequence. The axis that is making the homing is stopped.

4.3 *UsAxesManualJog*

Manual movimentation Axes

4.3.1 Bool AbsoluteRelativeJog

Property Read Write

Get or Set absolute/relative movimentation

False	Absolute
True	Relative

4.3.2 Bool ExternalJogActivated

Property Read Write

False	External Axes selector Disabled
True	External Axes selector Enabled

4.3.3 Int HandWheelMultiplier

Property Read Write

Handwheel multiplier read or set x1 x10 x100 x1000

4.3.4 Int SelectAxisForJog

Property Read Write

Read or set the axis for **MoveSelectAxisJog()**

4.3.5 Void MoveAxisJog(int AxisIndex,bool Direction)

JOG Axis to direction. The axis is selected by **AxisIndex**.

AxisIndex	Index of Axis
Direction	**True**/**False** Direction

WARNING!!!

Use **StopMove()** for stop movimentation

Example c#:

```
// Button down
private void button1_MouseDown(object sender, MouseEventArgs e)
{
   IsoNs.JogAxis.Jog(0,true);
}
// Button Up
private void button1_MouseUp(object sender, MouseEventArgs e)
{
   IsoNs.JogAxis.StopMove();
}
```

4.3.6 Void MoveAxisToTarget(double Target, Int32 AxisIndex)

Method that moves the axis to the position shown in Axis Target passed as doubles.
The unit of measurement of target position is set in the configuration.
If the CNC is in RELATIVE MOTION **AbssoluteRealtiveJog = True** target position is considered as the distance traveled from the point where it is the axis.
If the CNC is in ABSOLUTE MOTION **AbssoluteRealtiveJog = False** Target position is considered by the machine zero set for the axis movement

Target	Axis target position
AxisIndex	Index of Axis to be moved

WARNING!!!
Use **StopMove()** for stop movimentation

4.3.7 Void MoveSelectAxisJog(bool Direction)

JOG Axis to direction. The axis is selected **SelectAxisForJog**

Direction	**True**/**False** Direction

WARNING!!!
Use **StopMove()** for stop movimentation

4.3.8 Int32 ReadHandWheelMultiplier()

Return the Handwheel muliplier selected

4.3.9 Void SelectAxisHandMult(Int32 AxisIndex,Int32 HandWheelMultValue)

Select the HandWheel multiplier

AxisIndex	Index of Axis
HandWheelMultValue	Mulpilier value

The multiplier must have the following values

1	→	**x1**
10	→	**x1**
100	→	**x100**
1000	→	**x1000**

WARNING!!!
If you are using an electronic hand wheel with gearbox software, you should always use this method instead of the Property **SelectAxisForJog**.

4.3.10 Void SetPositionAxisShift(Int32 AxisIndex,double ShiftValue)

This method, moves the axis in Shift mode. The value ***ShiftValue*** is added at actual axis position (also if the axis is in moving). The entire position is reached during a time depending by parameter **TSHF_**...This value indicate the increment xTAU

AxisIndex	Index of Axis
ShiftValue	Value of Shift

Ex:
TAU=2 Ms
ShiftValue=100
TSHF_=10
The entire position is reached in 20 Ms

4.3.11 Void StopAllMove()

Stop all manual movements

4.3.12 EnumIsMoveAxes AxesIsMove ()

Return the Axes movment status

NoMove	No Axes Move
Move	Axes Move
CncInRun	Cn In Run Request not available

4.3.13 EnumMoveAxes MoveInterpolationAxes(StructMoveAxes AxesPar)

Motion Axes in interpolation

StructMoveAxes

```
public double Feed;  Feed Assi
public bool Absolute;  True Absolute motion
public double[] AxesValues;  Array Axes
public int HeadNumber;  Head -1 disable
public bool UseCurrentWorkOrigins;  True uses the current origins
public bool UseCurrentOffset;  True uses the current offet
```

Return:

CncInRun	Cn In Run Request not available
CncAlreadyInMovement	Axes already in movment
AllAxesNotReady	One ore more Axes are not ready (homing,enable)
NoAxesNumber	The AxesValues dimension is not equale to axes number
NoInsertBuffer	Movment not insert (wait axes move)
MoveAxesOk	Ok

4.4 UsAxesValue

Axes value

4.4.1 Int32 AxesResolution

Property Read
Axes value resolution setted in IsoUs.cfg

4.4.2 double CncTaskTime

Property Read
CNC sample time setting (millisecond)

4.4.3 Int32 FeedResolution

Property Read
Feed value resolution setted in IsoUs.cfg

4.4.4 double ReadDemandPosition(Int32 AxisIndex)

Reads the Demand position

AxisIndex	Index of Axis

4.4.5 double[] ReadAllDemandPosition()

Reads All Demand position of Axes

4.4.6 double ReadRealPosition(Int32 AxisIndex)

Reads the Real position

AxisIndex	Index of Axis

4.4.7 double[] ReadAllRealPosition()

Reads All Real position of Axes

4.4.8 double ReadRelativeDemandPosition(Int32 AxisIndex)

Reads the Relative (without offset) Demand position

AxisIndex	Index of Axis

4.4.9 double[] ReadAllRelativeDemandPosition()

Reads All Relative (without offset) Axes Demand position

4.4.10 double ReadRelativeRealPosition(Int32 AxisIndex)

Reads the Relative (without offset) Real position

AxisIndex	Index of Axis

4.4.11 double[] ReadAllRelativeRealPosition()

Reads All Relative (without offset) Axes Real position

4.5 *UsBreaKPoints*

It contains all the methods and properties to insert breakpoints. Breakpoints allow you to terminate the program at a desired point.
Once the program execution reaches the breakpoint that stops going to PAUSE.

4.5.1 Int32[] GetAllBreakPoint()

Return Array of Int32 containing all linee number of breakpoints inserted

4.5.2 Int32 InsertBreakPoint(Int32 LineNumber)

Insert a break point

LineNumber	Gcode Line Number

Return:
0 Break Point Ok
1 Break Point All ready inserted
2 Debug not activated

4.5.3 Bool IsBreakPoint(Int32 LineNumber)

Check if there is a Break point

LineNumber	Gcode Line Number

Return:

True	There is a BreakPoint
False	There is not a BreakPoint

4.5.4 Void RemoveAllBreakPoints()

Remove all break Points

4.5.5 Void RemoveBreakPoint(Int32 LineNumber)

LineNumber	Gcode Line Number

Return:
0 Break Point not inserted at line number
1 Break Point removed ok
2 Debug not activated

4.6 UsCalcTime

Manages all function for Gcode execution time calcolation

4.6.1 TimeSPan GetTotalTime

Property Read
Returns the time in second for Gcode execution

4.6.2 bool ExecuteCalcTime()

Start the cala time funtion
It is finished with event RunStop
Ritorna:
True Start Ok
False CNC not ready
GetTotalTime contains the timespan

4.7 UsCnErrors

It contains all the methods and properties to handle errors at the NC following an event **NotifyChanged**

4.7.1 bool **GetErrors(**out String[] **CnErrors,**out String[] **UsErrors)**

Return:
False no errors
True Errors
CnErrors array string of CNC errors(null none CNC errors)
UsErrors array string of IsoUs Errors (null none IsoUs errors)

4.7.2 void **ResetCnCAlarms**(**)**

Reste the CNC alarms

4.8 UsCnCMemory

Cnc Memory management.

4.8.1 Byte[] ReadArrayByteMemory(Int32 MemoryAddress,Int32 Length)

Reads an array of Bytes from CNC

MemoryAddress	Start memory Addres on CNC
Length	Number of bytes to read

Return:

Array of bytes

4.8.2 Int16[] ReadArrayInt16Memory(Int32 MemoryAddress,Int32 Length)

Reads an array of Ont16 from CNC

MemoryAddress	Start memory Addres on CNC
Length	Number of Int16 to read

Return:

Array of Int16

.

4.8.3 Int32[] ReadArrayInt32Memory(Int32 MemoryAddress,Int32 Length)

Reads an array of Int32 from CNC

MemoryAddress	Start memory Addres on CNC
Length	Number of Int32 to read

Return:

Array of Int32

4.8.4 void WriteArrayByteMemory(Int32 MemoryAddress, Byte[] DataValues)

Wrtite an array of Bytes in CNC memory

MemoryAddress	Start memory Addres on CNC
DataValues	Array of bytes to write

4.8.5 void WriteArrayInt16Memory(Int32 MemoryAddress, Int16[] DataValues)

Wrtite an array of Int16 in CNC memory

MemoryAddress	Start memory Addres on CNC
DataValues	Array of Int16 to write

4.8.6 void WriteArrayInt32Memory(Int32 MemoryAddress, Int32[] DataValues)

Wrtite an array of In32 in CNC memory

MemoryAddress	Start memory Addres on CNC
DataValues	Array of Int32 to write

4.9 *UsCncMFunctions*

It contains all the methods to execute the M in the CNC

4.9.1 Void Break_M_Function()

Break all M in execution on the CNC
The VTB Application must stop all M in execution
This method sets only the flag
ISOV1_STATUS_M_STOP

4.9.2 bool Execute_M_Cnc(Int32 Code_M_Function, Int32[] ParametersValue)

Run M On NC.

Code_M_Function	**M** code
ParametersValue	Int32 Array M paramaters (max number for parameters is defined IsoUs.cfg – default 10)

Return:
True M Run
False M not configured

Example c#:

```
private void GoM()
{
   Int32[] Param=new Int32[3];  // 3 parameters
   // assign the parameters values
   Param[0]=10;
   Param[1]=20;
   Param[2]=30;
   // Start M100
   if(MyIsoUs.UsCncMFunctions.Execute_M_Cnc(100, Param)==false)
      // error
}
```

4.9.3 Int32 ReadCnC_M_Parameters(Int32 ParameterNumber)

Read the M parameter on NC .This feature allows a complete interaction between the NC and the functions M PC application. If a function M activated the CN values must communicate to the PC is possible via this function

ParameterNumber	Parameter number on NC (0 to 10)

Return
Parameter value

4.9.4 bool WriteCnC_M_Parameters(Int32 ParameterNumber,Int32 ParameterValue)

Write a M par on NC

ParameterNumber	Parameter number
ParameterValue	Value

Return
True ok
False error
This method can also be used to write the variables defined by the compiler ... $ _PARAM1

4.10 UsCnCStatusWord

Questa classe contiene tutte le Property realtive allo stato del CN.

4.10.1 bool IsAbsoluteRealtiveMotion

Property Read
Absolute/relative motion

True	Relative Motion
False	Absolute motion (refrence to ZERO MACHINE)

4.10.2 bool IsStatusAxesMove

Property Read
Indicates the axes status movements

True	Axes Moviment
False	Axes stop

4.10.3 bool[] IsStatusEnableAxes

Property Read
Axes enable status.
Return a boolean array to length axes number
The relative array index contains the status enabled axis

True	Axis enabled
False	Axis disabled

4.10.4 bool IsStatusError

Property Read
Status NC error

True	NC in error
False	NC OK

4.10.5 bool IsStatusExternalOverride

Property Read
Indicates the exetrnal override status

True	External Override Enabled
False	External Override Disabled

4.10.6 bool[] IsStatuHomingAxes

Property Read
Axes homing status
Return a boolean array to length axes number
The relative array index contains the status homing axis

True	Axis homing OK
False	Axis homing NOT OK

4.10.7 bool IsStatusPause

Property Read
Indicates NC Stautus PAUSE

True	PAUSA NC
False	PAUSA terminated

4.10.8 bool IsStatusProbe

Property Read
Indicates the acquisition state (test after StartAcq())

True	Acquisition terminated
False	Acquisition in RUN

4.10.9 bool IsStatusRun Property di tipo Boolean- Read Only

Property Read
Indicates the NC statusRUN STOP

True	NC in RUN or PAUSE (check **IsStatusPause**)
False	NC in STOP

4.10.10 bool IsStatusStepMode

Property Read
Indicates the NC mode execution Gcode STEP MODE or NORMAL MODE

True	STEP MODE
False	NORMAL MODE

4.10.11 bool IsStatus_M_Execution

Property Read
Indicates if M to NC is in RUN

True	M in RUN
False	M Terminated

4.11 UsCompiler

Manages the Gcode Compilation

4.11.1 List<Compiler.MarkerCs> GetMarker

Property Read
Returns the LIST of marker defined in Gcode

4.11.2 List<Compiler.DimaArray> GetUsArray

Property Read
Returns the LIST of array defined in Gcode

4.11.3 List<string[]> GetUsDefine

Property Read
Returns the LIST of "define" defined in Gcode

4.11.4 List<string> GetUsFixedVariables

Reserved

4.11.5 List<string> GetUsVariables

Property Read
Returns the LIST of $ Variables defined in Gcode

4.11.6 string LastFileCompiled

Property Read
Returns the path of last file compilated

4.11.7 Int32 LastTotalCodeLoaded

Property Read
Returns the number of bytes loaded last time

4.11.8 Int32 TotalLinesCompiled

Property Read
Returns the number of lines compiled

4.11.9 UsWork.IsoUs.UsErrorCompiler[] CompileGcode(string Gcode,bool CanLoadCode, out Int32 TotalLines)

Compile Gcode from Text

Gcode	Text of Gcode (must be UPPERCASE)
CanLoadCode	True automatic load in memory
TotaLines	Returns the number of lines compiled

Return:
Array UsErrorCmpiler, or **null** if none errors

4.11.10 UsWork.IsoUs.UsErrorCompiler[] CompileGcodeFromBlock(string GcodePathFile,bool CanLoadCode, out Int32 TotalLines)

Compile Gcode from Block Execution

GcodePathFile	Path Gcode
CanLoadCode	True automatic load in memory
TotaLines	Returns the number of lines compiled

Return:
Array UsErrorCmpiler, or **null** if none errors

4.11.11 UsWork.IsoUs.UsErrorCompiler[] CompileGcodeFromFile(string GcodePathFile,bool CanLoadCode, out Int32 TotalLines)

Compile Gcode from file

GcodePathFile	Gcode Path
CanLoadCode	True automatic load in memory
TotalLines	Returns the number of lines compiled

Return**:**

Array UsErrorCmpiler, or **null** if none errors

WARNING This Function is Deprectaed Use **CompileGcodeFromPathFile**

4.11.12 UsWork.IsoUs.UsErrorCompiler[] CompileGcodeFromPathFile(string GcodePathFile,bool CanLoadCode, out Int32 TotalLines)

Compile Gcode from file

GcodePathFile	Gcode Path
CanLoadCode	True automatic load in memory
TotalLines	Returns the number of lines compiled **or -1 if file not found**

Return**:**

Array UsErrorCmpiler, or **null** if none errors

4.11.13 bool LoadCode()

Load in the memory the Gcode compiled

Return**:**

True Ok

False Error

4.12 UsConfig

It contains all the methods and properties that manage the configuration of the NC. The configuration is read from the file IsoUs.cfg.

4.13 UsGcodeRun

This class manages the execution of the ISO program

4.13.1 double Feed

Property Read Write
Reads or Set the Gcode **FEED**
Normally the **FEED** is setted by Gcode **F** function

4.13.2 bool SelectStepMode

Property Read Write
Get/Set the run STEP MODE Gcode

True	Step Mode enabled
False	Normal run

4.13.3 bool BackupCode()

BackUp the last code compiled.
This can be recovered by **RestoreGcode()**
BackUp avoids to recompile a Gcode file
Return**:**
True Ok
False BackUp non effettuato

4.13.4 bool ExecuteGcode(Int32 NlineStart)

Executes the Gcode loaded

NlineStart Number of Line of Start
if **NlineStart >0**, it means RESUME FROM BLOCK and the **MGOBLOCK** is performed

Return**:**

True	Ok
False	Error

4.13.5 Void ExecuteGcodeFromMarker(Int32[] AddressMarker,Double[] ValuesMarker)

Executes the Gcode when the conditions of the indicated marker.

MARKER

Markers are placed in the part of the normal variables.
IsoNs Gcode can resume when these variables have reached a certain value.
*In advanced programming with the use of VARIABLES and LOOP CYCLE, the recovery from the number of lines is not sufficient, since the positions axes can be increased from the value of variables that are detreminate a loop LOOP. Using markers, you can resume the Gcode by the value of one of these and not by line number, so it is possible to discriminate the resumption of cycles inside LOOP.**AddressMarker*** *Array di Int32 che contiene l' indirizzo fisico in memoria delle variabili MARKER recuperabile con la Property GetMarker, la quale ritorna un Lista di tipo* Compiler.MarkerCs

AddressMarker Array of Int32 that contains the memory address of Markers get by Compiler.MarkerCs

ValuesMarker Value must have each marker so that the restart condition is satisfied

Ex:
In the following example defines a variable Marker named **$INC**.

```
MARKER $INC NUMBER OF LOOP
$VAR=0
$INC=0
```

```
F5
G1X0Y0
LOOP 10
        $INC=$INC+1
        G1X200
        $VAR=$VAR+50
        G0X0Y[$VAR]
END_LOOP
```

You can then enable the resumption of the Gcode when the variable $ INC (the MARKER) assumes a certain value.

```
    private void GoMarker()
    {
      List<Compiler.MarkerCs> MyMarker = MyUsIso.UsCompiler.GetMarker; // Read Markers
      Int32[] AddrMarker = new Int32[MyMarker.Count];
      for (int n = 0; n < MyMarker.Count; n++)            // Copy address
        AddrMarker[n] = MyMarker[n].AddrVar;
      Int32[] ValMarker = new Int32[1]; // insert the marker value ($INC)
      ValMarker[0] = 5;                          // recovery after 5 cycles
      MyUsIso.UsGocodeRun.ExecuteGcodeFromMarker(AddrMarker, ValMarker); // Start
    }
```

4.13.6 Void ExecuteGcodeFromMarkerAndLine(Int32[] AddressMarker,Double[] ValuesMarker,Int32 LineNumber)

Equal to **ExecuteGcodeFromMarker** but the LineNumber condition must be satisfied

LineNumber Number of line

4.13.7 bool ExecuteGcodeScript(String GcodeScript, out UsWork.IsoUs.UsErrorCompiler[] Errors)

Execute Script code

A script behaves differently from a normal Gcode, as this does not handle a full movement of the axes but only G0 and G1 and can also be run at when the CN is in a state of PAUSE

GcodeScript Text of IsoUs Gcode Script

Errors **UsWork.IsoUs.UsErrorCompiler[]** Errors

Return**:**

True Ok

False Error

Example C#

```
    private void Script()
    {
      UsWork.IsoUs.UsErrorCompiler[] _Errors;
      if(!MyUsIso.UsGcodeRun.ExecuteGcodeScript("F5G1X100Y100,Z100\nG4F2\nX0Y0Z0", out _Errors))
      {
        if(Errors!=null)
        {
          for (int n = 0; n < Errors.Length; n++) // check errors
            Label1.Text += "E:" + Errors [n].Nline + " " + Errors [n].ErrorType;
        }
        else
          // execution Error
      }
      // Ok
    }
```

4.13.8 Void **PauseGcode()**

Pause Gcode
Is performed **MPAUSE** configured
From a state of PAUSE you can share the exact point where it stopped working by invoking the method **ExecuteGcode(0)**.
Is performed **MGOPAUSE** configured

4.13.9 bool **RestoreGcode()**

Recovery the code saved with **BackUpGcode()**
This is loaded in memory redy to work
Return**:**
True Ok
False Restore Error

4.13.10 Void **StopGcode()**

Force STOP Gcode
Is performed **MSTOP**configured

4.13.11 Int32 **WorkLineReal()**

Read Real line in esecution. Can also use event **WorkLineRealChanged**
It can be used when the CN is in STOP after an alarm to get the precise line of interruption of the Gcode.

4.14 UsGcodeVariables

This class provides access to the read and write variables ISONS Gcode.
Variables accessible are those of general type $.
It is necessary that the Gcode has been compiled.
The following variables are always at a fixed address

Name	Address	Name	Address
$_PARM_1	0	$_PARM_6	5
$_PARM_2	1	$_PARM_7	6
$_PARM_3	2	$_PARM_8	7
$_PARM_4	3	$_PARM_9	8
$_PARM_5	4	$_PARM_10	9

The above variables can be read and written also before the Gcode compilation
The possibility of writing the variables of the Gcode allows you to configure the processing cycle in a parametric mode.

4.14.1 List<string> GetAllVariablesName

Property Read
Get all Variable configured
Return:
The list of string with Variables Name without $

4.14.2 Int32 GetVariableAddress(String VariableName)

Get variable Address
VariableName Variable name without $
Return:
>=0 Variable Address
-1 Variable not found

4.14.3 String GetVariableName(Int32 VariableAddress)

Get variable Name
VariableAddress Variable Address
Return:
Name Variable name without $
“” Variable not found

4.14.4 double ReadVariableByAddress(Int32 VariableAddress)

Read Variable by Address
VariableAddress Variable Address
Return:
Value double
Exception variable not found

4.14.5 double ReadVariableByName(string VariableName)

Read Variable by Name
VariableName Variable name without $
Retrun:
Value double
Exception variable not found

4.14.6 bool WriteVariableByAddress(Int32 VariableAddress,double VariableValue)

Write Variable by Address

VariableAddress	Variable Address
VariableValue	Value

Return:

True	Ok
False	Variable not found

4.14.7 bool WriteVariableByName(string VariableName,double VariableValue)

Write Variable by Name

VariableName	Variable name without $
VariableValue	Value

Return:

True	Ok
False	Variable not found

4.15 UsInputOutput

It contains all the methods and properties to manage all the resources of the NC Input Output. The management of these resources depends on the type of CN used and the hardware configuration.

4.15.1 bool ReadDigitalInput(Int32 InputNumber)

Read Digital Input

InputNumber	Digital Input from 0 to 255

Return:

True	ON
False	OFF

4.15.2 bool ReadDigitalOutput(Int32 OutputNumber)

Read Digital Output

OutputNumber	Digital Output from 0 to 255

Return:

True	ON
False	OFF

4.15.3 Int32 ReadGroupDigitalInputs(Int32 Group)

Read a Group of Digital Inputs

Group	Group from 0 to 7 (0 first 32 digital inputs)

Return:

32 bit Digital Inputs

Bit Set	ON
Bit Reset	OFF

Example c#:

```
private void TestInput()
{
   // Read first group
   Int32 Group = MyUsIso.UsInputOutput.ReadGroupDigitalInputsg(0);
   // Test input1 and 5
   if((Group & 1)==1 && (Group & 16)==16)
      // ON
}
```

4.15.4 Int32 ReadGroupDigitalOutputs(Int32 Group)

Read a Group of Digital Outputs

Group	Group from 0 to 7 (0 first 32 digital outputs)

Return:

32 bit Digital Outputs

Bit Set	ON
Bit Reset	OFF

4.15.5 void WriteDigitalOutputs(Int32 OutputNumber,bool OutputState)

Write a Digital Outputs

OutputNumber	Digital Output from 0 to 255
OutputState	State **1** ON **0** OFF

4.16 UsLogFile

Log file management

4.16.1 string PathUsLog

Property Read Write

Set or Get the Path of Log file

4.17 UsMachineParameters

This class manages all parameters of CN.
All machine parameters are stored in the file and must be IsoUs.cfg

4.17.1 UsWork.UsMachineParametersCs.AxesVisType AxesValueMode

Property Read Write
Get or Set the type of read axes values:

DEMAND_POSITION	Demand Position
DEVIATION_POSITION	Following error
DISABLE	only DEMAND_POSITION
REAL_POSITION	Real Position

4.17.2 List<ComSynk.ParametriMacchina> MyParameterList

Property Read
Return the List of machine parameters

4.17.3 Int32 NumberOfParameters

Property Read
Return the number of machine parameters

4.17.4 String[] ParametersGroups

Property Read
Return the GROUP configured (ex: General,Axis X etc.)

4.17.5 String PathUsCfg

Property Read
Return the IsoUs.cfg Path

4.17.6 Void DownLoadParamaters()

Reads all the configuration parameters IsoUs.cfg and saves them in preparing an internal list to read the other of these methods.
<u>This should be in every case the first method to be invoked before the management parameters.</u>

4.17.7 bool GetParameterDataByIndex(Int32 ParameterIndex, out String ParameterName,out String ParameterDescr,out String ParameterGroup,out Int32 ParameterValue,out Int32 ParameterAddress)

Get a parameter from Index

ParameterIndex	Index of parameter in the internal List
ParameterName	Return Parameter Name
ParameterDescr	Return Parameter Description
ParameterGroup	Return Parameter Group
ParameterValue	Return Parameter Value
ParameterAddress	Return Parameter Address on CNC

Return:
True Ok
False Paramater not found or **DownLoadPara()** not performed

4.17.8 bool GetParameterDataByName (String ParameterName, out Int32 ParameterIndex,out String ParameterDescr,out String ParameterGroup,out Int32 ParameterValue,out Int32 ParameterAddress)

Get a parameter from Name

ParameterName	Parameter Name (CASE Sensitive)
ParameterIndex	Return Parameter Index
ParameterDescr	Return Parameter Description
ParameterGroup	Return Parameter Group
ParameterValue	Return Parameter Value
ParameterAddress	Return Parameter Address on CNC

Return:
True Ok
False Paramater not found or **DownLoadPara()** not performed

4.17.9 bool GetParameterExtendedData(Int32 ParameterIndex, out Int32 Transform,out Int32 MinValue,out Int32 MaxValue,out Int32 PassWordLevel,out Int32 Familiy, out Int32 AxisIndex, out List<ComSynk.EnumCs> Enums)

Get a parameter extend data from Index

ParameterIndex	Index of parameter in the internal List
Transform	Return Parameter Transform
MinValue	Return Parameter Minimum Value
MaxValue	Return Parameter Maximum Value
PassWordLevel	Return Parameter Password Level
Family	Return Parameter Familiy
AxisIndex	Return Parameter Axis Index (-1 none Axis)
Enums	Return Parameter ENUMERATIVE DESCRIPTION (if present)

Return:
True Ok
False Paramater not found or **DownLoadPara()** not performed

4.17.10 bool GetParameterValueByIndex(Int32 ParameterIndex,out Int32 ParameterValue)

Get a parameter from Index

ParameterIndex	Index of parameter in the internal List
ParameterValue	Return Parameter Value

Return:
True Ok
False Paramater not found or **DownLoadPara()** not performed

4.17.11 bool GetParameterValueByName (String ParameterName,out Int32 ParameterValue)

Get a parameter from Name

ParameterName	Parameter Name (CASE Sensitive)
ParameterValue	Return Parameter Value

Return:
True Ok
False Paramater not found or **DownLoadPara()** not performed

4.17.12 bool RestoreUsCfgBackUp()

Restore thye BackUp copy
Return:
True Restore Ok
False Error

4.17.13 Void SaveUsCfg()

Save in the **IsoUs.cfg** the parameter internal list
Before, a copy of backup is made (use **RestoreUsCfgBackUp()**)

4.17.14 Void SendAllParameters()

Send all Parameters to CNC
Some parameters are available only after **UpdateParameters()**

4.17.15 Void UpdateParameters()

Update the parameters
Call it after **SendAllParameters()**

4.17.16 bool WriteParametersByIndex(Int32 ParameterIndex, Int32 ParameterValue, bool WriteInCnC)

Write a parameter by Index

ParameterIndex	Index of parameter in the internal List
ParameterValue	Parameter Value
WriteInCnC	if **True** the Parameter is send to CNC (the **UpdateParameters() is not called**)

Return:

True	Ok
False	Error

4.17.17 bool WriteParametersByName(string ParameterName, Int32 ParameterValue, bool WriteInCnC)

Write a parameter by Name

ParameterName	Parameter Name
ParameterValue	Parameter Value
WriteInCnC	if **True** the Parameter is send to CNC (the **UpdateParameters() is not called**)

Return:

True	Ok
False	Error

4.18 UsMHMfunctions

It contains all the methods that handle the compilation and Property of M and HM, which are run at a PC (not inside the CN)
These can then be retrieved by the method ExecuteMtoCn.GoMtoCn (....)

4.18.1 bool GenerateHMFunction (Int32 HMFunctionNumebr, String GCode,out UsWork.IsoUs.UsErrorCompiler[] Errors)

4.18.2 bool GenerateMFunction (Int32 MFunctionNumebr, String GCode,out UsWork.IsoUs.UsErrorCompiler[] Errors)

This method allows you to fill out a HM or M function and save it automatically in the configuration folder.

..FunctionNumber M or HM number
GCode String Code ISO
Errors Array Error compiler
Return:
True M/HM Ok
False Error

4.19 *UsOffsetAndOrigins*

This class manages the capabilities of the MANUAL WORK ORIGIN.
IsoUs manipulate an array of 256 different positions for WORK ORIGIN. The positions are indexed by GCODE instruction **USER_ZERO Index Value**.
This function is the same of GCODE G94

4.19.1 List<ComSynk.FileZeri> GetAllOriginsByFile

Property Read
Get all origins setted in the dafault file
Return:
List of 9 string position that contains the values of origin for single Axis
The List position is the Index of Origin

4.19.2 bool HeadOffsetSuspend

Property Read Write
Suspend or Resume the Heads Offset selected by **Hn**

False	Suspend (As G87)
True	Resume (As G88)

4.19.3 Int32 IndexOffset

Property Read Write
Get or Set the Index OFFSET
Value from 0 to 255

4.19.4 Int32 IndexOrigin

Property Read Write
Get or Set the Index ORIGINS
Value from 0 to 255

4.19.5 bool IsOffset

Property Read
Read if the OFFSET is activated

False	Disabled
True	Enabled

4.19.6 bool IsOrigins

Property Read
Read if the ORIGINS are activated

False	Disabled
True	Enabled

4.19.7 bool OffsetSuspend

Property Read Write
Suspend or Resume the Offset selected by **G93**

False	Suspend (As G96)
True	Resume (As G97)

4.19.8 bool OriginsSuspend

Property Read Write
Suspend or Resume the Origins selected by **G94**

False	Suspend (As G98)
True	Resume (As G99)

4.19.9 Int32[] SingleAxisIndexOrigins

Property Read Write
Get or Set the index origin fo the single Axis
Array Int32 that contains the Index Origins for Axis

4.19.10 Void ActivateOriginFile()

Activates the Origins by file ZERI.VAL (default File)

4.19.11 Int32 AxesOffsetDisable(Int32 Index)

Disable the Offset at Index

Index	Offset Index

Return:

1	Error
0	Ok

4.19.12 Int32 AxesOriginDisable(Int32 Index)

Disable the Origin at Index

Index	Origin Index

Return:

1	Error
0	Ok

4.19.13 double[] GetCurrentOffset(Int32 Index)

Get the Current offset at Index

Index	Offset Index

Return:
Array of Axes Offset position

4.19.14 double[] GetCurrentOffsetSetted()

Get the Offset at current Index
Return:
Array of Axes Offset position

4.19.15 double[] GetCurrentOrigins(Int32 Index)

Get the Current Origin at Index

Index	Origin Index

Return:
Array of Axes Origin position

4.19.16 double[] GetCurrentOriginsSetted()

Get the Origin at current Index
Return:
Array of Axes Origin position

4.19.17 double[] GetOriginsFromUsFile(Int32 Index)

Get the Origins at Index from default file ZERI.VAL

Index	Origin Index

Return:
Array of Axes Origin position

4.19.18 Int32 SetAxesOffsetToCurrentPosition(Int32 OffsetIndex)

Set the OFFSET at Index to current axes position

OffsetIndex	Offset Index

The OFFSET is automatically enabled

Return:

1	Error
0	Ok

4.19.19 Int32 SetAxesOffsetToPosition(double[] OffsetValue, Int32 OffsetIndex)

Set the OFFSET at Index by values

OffsetValue	array of Offset Values for Axis
OffsetIndex	Offset Index

The OFFSET is automatically enabled

Return:

1	Error
0	Ok

4.19.20 Int32 SetAxesOriginsToCurrentPosition(Int32 OriginIndex)

Set the ORIGIN at Index to current axes position

OriginIndex	Origin Index

The ORIGIN is automatically enabled

Return:

1	Error
0	Ok

4.19.21 Int32 SetAxesOriginsToPosition(double[] OriginsValue, Int32 OringinIndex)

Set the ORIGIN at Index by values

OriginsValue	array of Origin Values for Axis
OriginIndex	Origin Index

The ORIGIN is automatically enabled

Return:

1	Errore
0	Ok

4.19.22 Int32 SetAxisOffsetToCurrentPosition(Int32 OffsetIndex, Int32 AxisIndex)

Set the OFFSET at Index by current position for single Axis

OffsetIndex	Offset Index
AxisIndex	Axis Index

The OFFSET is automatically enabled

Return:

1	Error
0	Ok

4.19.23 Int32 SetAxisOffsetToPosition(double OffsetValue,Int32 OffsetIndex, Int32 AxisIndex)

Set the OFFSET at Index by value for single Axis

OffsetValue	Offset Value
OffsetIndex	Offset Index
AxisIndex	Axis Index

The OFFSET is automatically enabled

Return:

1	Error
0	Ok

4.19.24 Int32 SetAxisOriginToCurrentPosition(Int32 OriginIndex, Int32 AxisIndex)

Set the ORIGIN at Index by current position for single Axis

OriginIndex	Origin Index
AxisIndex	Axis Index

The ORIGIN is automatically enabled

Return:

1	Error
0	Ok

4.19.25 Int32 SetAxisOriginToPosition(double OriginValue,Int32 OriginIndex, Int32 AxisIndex)

Set the ORIGIN at Index by value for single Axis

OrigintValue	Origin Value
OriginIndex	Origin Index
AxisIndex	Axis Index

The ORIGIN is automatically enabled

Return:

1	Error
0	Ok
0	Ok

4.19.26 bool IsoG43FromTable(Int32 Kmode,Int32 Axis, bool Direction)

Enable G43 from Tool Table

Kmode	See the K parameter G43 function
Axis	Axis Index
Direction	Correction direction (**false** Negative – **true** Positive)

Return:

True	Ok
False	Error (CNC in Run or Axis not configured)

4.19.27 bool IsoG43FromLen(Double ToolLen,Int32 Kmode,Int32 Axis, bool Direction)

Enable G43 from Tool Len

ToolLen	Tool Len
Kmode	See the K parameter G43 function
Axis	Axis Index
Direction	Correction direction (**false** Negative – **true** Positive)

Return:

True	Ok
False	Error (CNC in Run or Axis not configured)

4.19.28 bool IsoG44(Int32 G44Ext)

Disable G43

G44Ext	See command **G43.0 G43.1**

Return:

True	Ok
False	Error (CNC in Run or Axis not configured)

4.19.29 bool IsoG43State()

Read the G43 state

Return:

True	G43 Enabled
False	G43 Disabled

4.20 UsOverrideFeed

This class manages the OVERRIDE FEED

.

4.20.1 bool ExternalOverrideFeed

Property Read Write

Enable/Disable External Override

True	Enabled
False	Disabled

4.20.2 Int32 OverrideFeedAxes

Property Read Write

Get or Set the internal override value

Value from 0 to 100 %

4.21 UsPassWordManagement

IsoUs PassWord managaement

4.21.1 bool ResetUsPassWord(Int32 PassWordLevel)

Reset the current Password to defualt value

PassWordLevel	PassWord level to Reset from 0 to 2

Defualt value:

Level 0	→	684618
Level 1	→	684619
Level 2	→	684620

Return:

True	Ok
False	Error

4.21.2 bool SetUsNewPassWord(String NewPassWord,Int32 PassWordLevel)

Set the new PassWord at level

NewPassWord	New Password
PassWordLevel	PassWord level to Set from 0 to 2

Return:

True	Ok
False	Error

4.21.3 bool TestUsLevelPassWordBlock(Int32 PassWordLevel)

Check the PassWord level if locked or Unlocked

PassWordLevel	PassWord level to check from 0 to 2

Return:

True	Unlocked
False	Locked

4.21.4 bool TestUsPassWord(String PassWord,Int32 PassWordLevel)

Check the PassWord at level

PassWord	PassWord to check
PassWordLevel	PassWord level to Check from 0 to 2

Return:

True	PassWord Ok
False	PassWord Error

4.22 *UsPositioners*

Positioners class manager
The positioners must be enabled in VTB application

4.22.1 bool AllAxesPositionerReady

Property Read
Return the state of all positioners (Homing and Enable)

True	Ready
False	Not Ready

4.22.2 Int32 NumberOfPositioners

Property Read
Return the number of positioners declared in VTB application

4.22.3 bool EnablePositioner(Int32 PositionerIndex,bool EnableState)

Enable/Disable Positioner

PositionerIndex	Positioner Index
EnableState	**True** Enable **False** Disable

Return:

True	Ok
False	Positioner Error

4.22.4 bool IsAlarm(Int32 PositionerIndex,out bool Status)

Reads the Alarm status

PositionerIndex	Positioner Index
Status	**True** Alarm **False** Ok

Return:

True	Ok
False	Positioner Error

4.22.5 bool IsEnabled(Int32 PositionerIndex,out bool Status)

Reads the Enable status

PositionerIndex	Positioner Index
Status	**True** Enabled **False** Disabled

Return:

True	Ok
False	Positioner Error

4.22.6 bool IsHoming(Int32 PositionerIndex,out bool Status)

Reads the Homing status

PositionerIndex	Positioner Index
Status	**True** Homing Ok **False** Homing not made

Return:

True	Ok
False	Positioner Error

4.22.7 bool IsMoving(Int32 PositionerIndex,out bool Status)

Reads the Movement status

PositionerIndex	Positioner Index
Status	**True** Movement **False** Stop

Return:

True	Ok
False	Positioner Error

4.22.8 bool Preset(Int32 PositionerIndex, Int32 PresetValue, out bool Status)

Preset Axis value

PositionerIndex	Positioner Index
PresetValue	Preset Valòue

Status **True** Preset Ok **False** Preset Error

Return:

True	Ok
False	Positioner Error

4.22.9 bool ReadDemandPosition(Int32 PositionerIndex, out Int32 PositionValue)

Reads Demand Position

PositionerIndex	Positioner Index
PositionValue	Demand Position value

Return:

True	Ok
False	Positioner Error

4.22.10 bool ReadRealPosition(Int32 PositionerIndex, out Int32 PositionValue)

Reads Real Position

PositionerIndex	Positioner Index
PositionValue	Real Position value

Return:

True	Ok
False	Positioner Error

4.22.11 bool SetOffsetValue(Int32 PositionerIndex, Int32 OffsetValue)

Set OFFSET position

PositionerIndex	Positioner Index
OffsetValue	OFFSET value

Return:

True	Ok
False	Positioner Error

4.22.12 bool SetVelocity(Int32 PositionerIndex, Int32 VelocityValue)

Set FEED

PositionerIndex	Positioner Index
VelocityValue	FEED value

Return:

True	Ok
False	Positioner Error

4.22.13 bool StartHoming(Int32 PositionerIndex)

Start homing

Read StatusWord for check end Homing

PositionerIndex	Positioner Index

Return:

True	Ok
False	Positioner Error

4.22.14 bool **StartPositionTarget(**Int32 **PositionerIndex,** Int32 **PositionValue,** bool **AbsoluteValue)**

Start positioner at Target

PositionerIndex	Positioner Index
PositionValue	Target Value
AbsoluteValue	**True** Target Absolute Value **False** Target Realtive Value

Return:

True	Ok
False	Positioner Error

4.22.15 bool **StatusWord(**Int32 **PositionerIndex,** Out Int32 **Status)**

Reads StatusWord

PositionerIndex	Positioner Index
Status	StatusWord bit mapped (bit 1 setted):

Bit 0	→	**Enable**
Bit 1	→	**Homing**
Bit 2	→	**Move**
Bit 3	→	**Allarm**

Return:

True	Ok
False	Positioner Error

4.22.16 bool **Stop(**Int32 **PositionerIndex)**

Stop the Positioner

PositionerIndex	Positioner Index

Return:

True	Ok
False	Positioner Error

4.23 UsRetrace

This class handles all the logic functionality of IsoNs Retrace.
The retrace and a mode useful for some types of machines. In practice, it scrolls the tool path on the piece with a kind of simulation, but with axes in motion. The path of the sliding mode occurs in both forward and reverse JOG.This allows you to choose how to RESTART POINT OF VIEW of any section even though this is not a starting point or end of an element. you can make a fresh start from any point of an arc or a straight line. With axes in motion, can have a real vision on the machine POINT OF RESTART.
Is performed **MGORETRACE** configured
Before using the functions of retrace, you must call the method InitRetrace()

4.23.1 Int32 GetLineRetrace

Property Read
Return the Gcode Line currently in execution in retrace mode.

4.23.2 Void ExecuteProg()

Start the Gcode from the current point where the axes are on the profile.
This method is invoked when the operator decided the point of restart of the profile.
Is performed **MGORETRACE** configured

4.23.3 bool GetPosAxisAtLine(out Int32 PosX, out Int32 Posy, Int32 Nline)

Reads the axes position at line number

Posx	Axis X Position
Posy	Axis Y Position
NLine	Gcode Line number

Return:

True	Ok
False	Line not found

4.23.4 Void GoLine(Int32 Nline)

Move axes to block the line indicated.
Need to skip parts of the tool path.

Nline	Gcode Line number to jump

4.23.5 Void InitRetrace()

Enable the feature RETRACE. This method must be called before using any of the Property and the other class methods retraced.
In practice prepares a list of all the Gcode simulation.

4.23.6 Void JogDown()

Move axis in the negative direction on the path PIECE.
The jog speed is set in the Gcode
For STOP Axes use **StopJog ()**

4.23.7 Void JogUp()

Move axis in the positive direction on the path PIECE.
The jog speed is set in the Gcode
For STOP Axes use **StopJog ()**

4.23.8 Void StopJog()

Stp Axes **JogUp()** or **JogDown()**

4.24 UsSimulation

Simulation Gcode

4.24.1 Void **ExecuteSimulation()**

The current Gcode loaded, will be simulated in the preview window

4.25 *UsSpindleManager*

This Class allows to manage all functions for Analog 0-10V Spindle Output.
The Spindle Output is configured in IsoUs by Machine Parameters Spindle Table
(See IsoUs Documentation)

4.25.1 Void **WriteSpindleSpeed(**Int32 **_Val)**

Write the value **_Val** in the Analog Output configured for Spindle
_Val is in **BIT**, therefore fo an output of **10V** on the Channels **0-15** the value must be **2047**, for the Channel **NGMEVO PWM 255** (max)

4.25.2 Int32 **ReadSpindleSpeed()**

Read the last value write with ***WriteSpindleSpeed()***

Return	0-2047 for the channels 0-15
	0-255 for the channel NGMEVO PWM

4.25.3 Void **WriteSpindleOw(**Int32 **_Val)**

Write the Override value for the Analog Output configured for Spindle
The value is between **0** (0%) and **1024** (100%).
WARNING
For this function, must be enabled the parameter **ENABLE_OW_SPEED** on **INTERNAL VIRTUAL**
The Minimum and Maximum value is restrained by parameter:

SPEED_OW_MIN	Minimum Value of Override
SPEED_OW_MAX	Maximum Value of Override

4.25.4 Int32 **ReadSpindleOw()**

Read the Overide value set
Return 0-1024

4.25.5 Int32 SpindleOwState

Property Read
Return the mode set of Spindle Override

0	**Disable**
1	**Managed by VTB**
2	**Managed by WriteSpindleOw()**

4.25.6 Int32 SpindleAnalogBitRes

Property Read
Return the BIT resolution set for the Spindle Analog Output

2048	**Channels 0-15**
255 (max)	**Channel NGMEVO PWM**

4.26 UsStaticVariables

This class handles files of variables saved in permanent memory. IsoNs allows you to save the contents of variables in a file so that it can be reloaded later. These files are also manageable Gcode.Le PERMANENT are all variables of type DOUBLE, and are included in a list. Ilist<Double>.
The file are saved in internal path of

4.26.1 String GetPathStaticFile

Property Read
Return the path of permanents variables.

4.26.2 List<double> GetStaticVariables

Property Read
Get the values of static variables.
Necessary use **LoadStaticFile** before

4.26.3 bool LoadStaticFile(String StaticFileName)

Load in memory the static file

StaticFileName	File name (Use GetPathStaticFile for get the Path)

Return:

True	Ok
False	File not found

4.26.4 Void SaveFile(String StaticFileName)

Save in the file the curren LIST get from **GetStaticVariables**.

StaticFileName	File Name (only name)

4.27 UsToolInfo

Tool informations

4.27.1 Int32 GetIndexAxisLengthEnabled

Property Read

Get the Index Axis where is enabled the Length correction

-1	No Axis

4.27.2 double GetToolDiameterSet

Property Read

Get the current Tool Diameter setted

4.27.3 bool GetToolDirection

Property Read

Get the Tool Length correction direction

True	Positive
False	Negative

4.27.4 double GetToolLengthSet

Property Read

Get the Length correction value

4.27.5 bool IsToolLengthActivated

Property Read

Get if enabled Tool Length correction

True	Enabled
False	Disabled

4.28 UsToolsHeadsTable

It contains all the methods and properties related to the management of the tool **HEADS** and the parameters of the **TOOLS** table.

4.28.1 Int32 GetNumberOfHeads

Property Read
Get the number of Heads configured

4.28.2 Int32 GetNumberOfToolsTable

Property Read
Get the number of Tools configured

4.28.3 Int32 SelectHEad

Property Read Write
Get or Set an **HEAD**.
The value of SET must be between 0 and the maximum number of HEAD included in the configuration. An exception is generated if the value is not within this range.
Returns -1 if no **HEAD** is selected
This function is equal to **Hn** in Gcode

4.28.4 Int32 SelectTollTable

Property Read Write
Get or Set a **TOOL TABLE**.
The value of SET must be between 0 and the maximum number of TABLES included in the configuration tool. An exception is generated if the value is not within this range.
Returns -1 if no TABLE is selected
This function is equal to **Tn** in Gcode

4.28.5 double GetHeadParameter(Int32 ParameterIndex)

Reads the parameter specified in **ParameterIndex** from the **HEAD** table selected.
ParameterIndex Index of parameter to selected head
Return:
Parameter Value

4.28.6 double GetToolParamater(Int32 ParameterIndex)

Reads the parameter specified in **ParameterIndex** from the **TOOL** table selected.
ParameterIndex Index of parameter to selected Tool
Return:
Parameter Value

4.28.7 Void SaveToolTabelParameters()

Save the parameters of the tool table in memory (written with WriteParTab) in the configuration on disk permanently

4.28.8 Void WriteHeadParameter(double Value,Int32 HeadIndex, Int32 ParameterIndex)

Write a parameter to Head

Value	Parameter value
HeadIndex	Head Index
ParameterIndex	Parameter index

4.28.9 Void WriteToolParameter(double Value,Int32 TableIndex, Int32 ParameterIndex)

Write a parameter to Tool

Value	Parameter value
TabelIndex	Tool Index
ParameterIndex	Parameter index

4.29 *UsGenericVariables*

This class handles variables UserGenric for data exchange with the NC.
These variables are of type Int32, and can be used to read / write values of using them for general data exchange between PC applcazione CN VTB.
The number of variables is 30

4.29.1 Int32 ReadUserCnCVariable(Int32 UserAddress)

Reads a User Generic Variable from CNC

UserAddress	User Address from 0 to 29

Return:
User Value

4.29.2 Void WriteUserCnCVariable(Int32 UserAddress,Int32 UserValue)

Writes a User Generic Variable in the CNC

UserAddress	User Address from 0 to 29
UserValue	User Value

4.30 Us2ndLimitsManager

Racchiude tutti i metodi e proprietà che gestiscono i secondi limiti assi

4.30.1 bool Is2ndLimits

Property Read
Return:
True 2nd Limits activated
False Primary Limits activated

4.30.2 bool Activated2ndLimits()

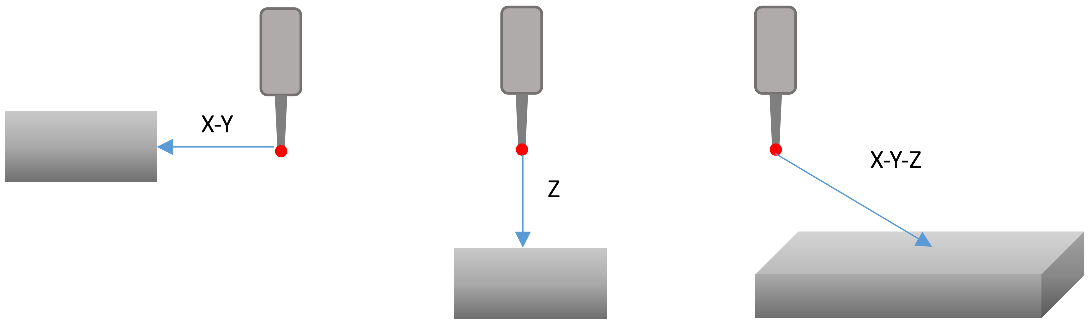
Activate the **2nd** Software Limits
Return:
True – Ok
False – CnC in Run

4.30.3 bool Reset2ndLimits()

Deactivates the **2nd** Limits and activates the **PRIMARY** software Limits
Ritorna:
True – Ok
False – CnC in Run

4.31 UsProbe

This class manages the capabilities of Acquisitions sensor boards. Acquisitions by The sensor requires that the axes are moved to a position up and be detected by this sensor.
The axes can be moved at one time, however, can move only one axis in the direction of the probe Acquisition .



4.31.1 Double[] GetAxesValue

Property Read
Read acquisition positions axes
After acquisition executed you can use this prpoerty for read all positions axes
Return:
Double[] Array axes positions at sensor

4.31.2 bool StartProbeAcquisition(double[] AxesValue)

Start Acquisition at postion
When the sensor detected the piece, the **AcqusitionExecuteChanged** event is performed.
Otherwise **NotifyChanged** event is performed if axes reach the position target.
Is possible test StatusCn **IsStatusProbe**.

AxesValue	Array for Target positions.

Return:

True	Acquisition Started
False	Error Axes position array

WARNING
Before run **StartProbeAcquistion** you must set the RUN NC by following instructions
MyMaster.GetCommand.RunProg();
At end (normal end or error) **StartProbeAcquistion** you must set the STOP NC by following instructions
MyMaster.GetCommand.StopProg();

4.32 UsMultiProcess

This class manages the MultiProcess capability of IsoUs.
All CN manage of PC are visible by this class

4.32.1 MProcessResult GcodeLoadFromFile(Int ProcessNumber, string GcodePat, Int LoadInEditor)

Load a Gcode from file

ProcessNumber	Process Number 0 to 7
GcodePath	Gcode Path
LoadInEditor	if **True** the will be load in Editor

Return:

MProcessResult.MprocessOk	Operation OK
MProcessResult.MprocessNumberError	Process Number not found
MProcessResult.MprocessFileNotFound	Gcode not found
MProcessResult.MprocessCNCinRun	Error CNC in run
MProcessResult. MprocessGcodeCompileErrore	Gcode Compile Error
MProcessResult. MprocessUsEditorNotFound	**LoadInEditor** Error.

4.32.2 List<IsoUs> ActiveProcess

This List contains all UsWork components for all CN active in the current configuration
By it is possible to accesa t all UsWork

ActiveProcess[0]	**Process 0**
ActiveProcess[1]	**Process 1**

Etc.

4.33 *UsCmdManager*

This class allows to manage teh **CMD** of IsoUs.
The CMD are the Binary file saved in the local folder ***_CmdBinary*** with extension ***.USB***
These are the Gcode Files generally compiled with **PlugIn MHM**.
The CMD can be executed when the CN is in **STOP** or in **PAUSE** (as **SCRIPT**).
The CMDs are functions that can customize the machine.
Can be passed up to **10 Parameters** (Double) that can be read in the Gcode file by:
$[X18…X27]

4.33.1 CmdResult RunCmd(string CmdName, int Cpu, double[] Parameters, bool CheckAxesReady)

Executes a CMD

CmdName	CMD name (Without extension)
Cpu	CPU Number
	0 Normal Run (Only CNC in STOP)
	1 Run as SCRIPT (CNC in STOP or PAUSE)
Parameters	Array of double read in the Gcode by:
	(The variable **$[X17]=1** indicate that a CMD is in execution)
	$[X18] Parameters[0]
	$[X19] Parameters[1]
	$[X20] Parameters[2]
	$[X21] Parameters[3]
	$[X22] Parameters[4]
	$[X23] Parameters[5]
	$[X24] Parameters[6]
	$[X25] Parameters[7]
	$[X26] Parameters[8]
	$[X27] Parameters[9]
CheckAxesReady	**False** The Axes Ready are not checked (HOME and ENABLE performed for all Axes)

Return:

CmdResult.CmdRunOk	CMD Executed
CmdResult.ErrorCmdNotFound	CMD file not found
CmdResult.ErrorCmdCpuInRun	CN in RUN
CmdResult.ErrorCmdParameters	More than 10 Parameters are passed
CmdResult. ErrprAxesNotReady	Axes not Ready (CheckAxesReady=True)

4.33.2 bool SaveCmdCode(string Cmd, bool CheckFile)

Allows to generate a CMD by the Gcode current compiled
The CMD is saved in ***_CmdBinary***

Cmd	CMD name (without extension)
CheckFile	**True** is checked if the CMD already exists (a Dialog Box will be showed)

Return:

True	CMD Generate OK
False	Error

4.33.3 string CurrentCmdInRun

Property Read
Return the CMD name in execution

4.33.4 bool IsRunCmd

Property Read
True CMD in RUN

4.34 UsTask

This class manage the **PARALLEL TASKS** of IsoUs.
There are Two Parallel Tasks TASK0 e TASK1 that can execute the Gcode concurrently to main process.
TASK 0 is refered to TASK 1 in Gcode Instructions
TASK 1 is refered to TASK 2 in Gcode Instructions

```
public enum UsTaskError
        {
            NoError,
            NoTask,
            AlreadyInRun,
            AlreadyInPause,
            NoTaskInRun,
            NoAddrVarValid,
            CmdPathNotFound,
            PrioriryOutOfRange,
            CmdAutoLoadNotSet,
            NoTypeRun,
        }
```

4.34.1 IsoUs.ErrorCompiler[] LoadGcode(string Gcode, int UsTask, out UsTaskError Error)

Load the Gcode in the Task

Gcode	Gcode String
UsTask	Task Number (0 or 1)
Out Error	UsTaskError
Return	IsoUs.ErrorCompiler[]

4.34.2 UsTaskError TaskRun(int UsTask)

Run Gcode in the Task

UsTask	Task Number (0 or 1)
Return	UsTaskError

4.34.3 UsTaskError TaskStop(int UsTask)

Stop Gcode in the Task

UsTask	Task Number (0 or 1)
Return	UsTaskError

4.34.4 UsTaskError TaskPause(int UsTask)

Pause Gcode in the Task

UsTask	Task Number (0 or 1)
Return	UsTaskError

4.34.5 UsTaskError TaskReadVar(int UsTask, int AddrVar, out double ValVar)

Read Variable in the Task

UsTask	Task Number (0 or 1)
AddrVar	Source Variable Address
Out ValVar	Variable value
Return	UsTaskError

4.34.6 UsTaskError TaskWriteVar(int UsTask, int AddrVar, double ValVar)

Write a variable in the Task

UsTask	Task Number (0 or 1)
AddrVar	Destination variable Address
ValVar	Source variable value
Return	UsTaskError

4.34.7 UsTaskError TaskStatus(int UsTask, out UsTaskStatus Status)

Read Task Status

UsTask	Task Number (0 or 1)
Out Status	Status
Return	UsTaskError

```
public enum UsTaskStatus
{
    Stop,
    Run,
    Pause,
}
```

4.34.8 UsTaskError TaskLoadCmd(int UsTask, String CmdName)

Load CMD in the TASK

UsTask	Task Number (0 or 1)
CmdName	CMD name without extension
Return	UsTaskError

4.34.9 UsTaskError TestTaskCmdAutoLoad(int UsTask)

Test if a CMD should be loaded automatically after a RUN gcode in the TASK
The Autoload CMD is defined in the IsoUs Environment file via SetTaskCmdAutolad
In the positive case, the CMD file is loaded and executed

UsTask	Task Number (0 or 1)
Return	UsTaskError

4.34.10 UsTaskError SetTaskCmdAutoLoad(int UsTask, String CmdName, bool IsAutoRun)

Set the CMD as Autoload in the IsoUs Environment file.
This is always executed / loaded on the Gcode Run

UsTask	Task Number (0 or 1)
CmdName	CMD name without extension
IsAutoRun	If True, the CMD is Executed automatically to RUN gcode, else only loaded
Return	UsTaskError

4.34.11 UsTaskError TaskSetPriority(int UsTask, int Priority)

Set priority Task
Lower values higher priority Default 0 (max)

UsTask	Task Number (0 or 1)
Priority	priority Task from 0 to 100
Return	UsTaskError

Index